

Federated Identity Management and Web Services Security

with IBM Tivoli Security Solutions

Introduction to Web services security
standards

Complete product architecture
and component discussion

Extensive federation
business scenario



Axel Buecker
Werner Filip
Heather Hinton
Heinz Peter Hippenstiel
Mark Hollin
Ray Neucom
Shane Weeden
Johan Westman



International Technical Support Organization

**Federated Identity Management and Web Services
Security with IBM Tivoli Security Solutions**

October 2005

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.

Second Edition (October 2005)

This edition applies to Version 6 of Tivoli Federated Identity Manager (product number 5724-L73) and to all subsequent releases and modifications until otherwise indicated in new editions. Various other related IBM and Tivoli products are mentioned in this book.

© Copyright International Business Machines Corporation 2004, 2005. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xiii
Trademarks	xiv
Preface	xv
The team that wrote this redbook	xvi
Become a published author	xviii
Comments welcome	xix
Part 1. Architecture and design	1
Chapter 1. Business context for identity federation	3
1.1 Federated identity	4
1.2 Business environment	5
1.2.1 Deconstruction of the enterprise	5
1.2.2 Enterprise re-aggregation	6
1.2.3 High-level example of a re-aggregated business	7
1.2.4 Business models for federated identity	9
1.2.5 The relationship - Trust and assurance	15
1.3 IT environment	17
1.3.1 The role of identity management	17
1.3.2 Dealing with identities	20
1.3.3 User life cycle management	23
1.3.4 Inter-enterprise application to application integration	25
1.3.5 Open standards	27
1.4 Conclusion	28
Chapter 2. Architecting an identity federation	31
2.1 Federation example	33
2.2 Federated identity management architecture	36
2.2.1 Background to federation	37
2.2.2 Architecture overview	38
2.2.3 Roles	41
2.2.4 Identity models	42
2.2.5 Identity attributes	45
2.2.6 Trust	49
2.2.7 Federation protocol	51
2.3 FIM standards and efforts	51
2.3.1 SSL/TSL	52
2.3.2 Security Assertion Markup Language (SAML)	52

2.3.3	Shibboleth	53
2.3.4	Liberty	54
2.3.5	WS-Federation	55
2.3.6	WS-Trust	56
2.3.7	WS-Security	57
2.3.8	WS-Provisioning	57
2.3.9	Selecting Federation standards	58
2.4	Federated single sign-on	59
2.4.1	Push and Pull SSO	60
2.4.2	Account linking	61
2.4.3	Where are you from (WAYF)	62
2.4.4	Session management and access rights	63
2.4.5	Logout	63
2.4.6	Credentials clean up	64
2.4.7	Global good-bye	64
2.4.8	Account de-linking	65
2.5	Web services security management	65
2.5.1	Web services	66
2.5.2	Web services security	68
2.5.3	Gateways	69
2.6	Federated identity provisioning	70
2.7	On demand security reference architecture	72
2.7.1	Policy management	73
2.7.2	Identity management	74
2.7.3	Key management	74
2.7.4	Credential exchange	74
2.7.5	Identity federation	75
2.7.6	Authorization	75
2.8	On demand integration reference architecture	75
2.8.1	Connectivity services	77
2.8.2	User interaction services	77
2.8.3	Application and information assets	78
2.8.4	Business application services	78
2.8.5	Partner services	78
2.8.6	Infrastructure services	79
2.9	Method for architecting secure solution	79
2.9.1	Implementation flow	80
2.9.2	Definition phase of a federated identity management solution	81
2.10	Conclusion	83
Chapter 3. Tivoli Federated Identity Manager architecture		85
3.1	Federated Identity Management functionality	86
3.2	Federation services	87

3.2.1	Point of contact (PoC)	89
3.2.2	Single sign-on protocol services (SPS)	91
3.2.3	Trust services	92
3.2.4	Key services (KESS)	96
3.2.5	Identity services	97
3.2.6	Authorization services	97
3.2.7	Provisioning services	98
3.2.8	Management Services	98
3.3	Federated single sign-on	100
3.3.1	Architecture overview	103
3.3.2	Trust in F-SSO	104
3.3.3	F-SSO protocol functionality	105
3.3.4	Integrating SSO with Access Manager for e-business	109
3.3.5	F-SSO approaches	110
3.3.6	InfoService	119
3.3.7	Specified level view of F-SSO architecture	120
3.4	Web services security management	121
3.4.1	Architecture overview	123
3.4.2	WS-Security	125
3.4.3	Web services Gateway or Firewall	126
3.4.4	WS-Trust	127
3.4.5	Authorization services (AS)	128
3.4.6	Web services security management architecture approach	128
3.5	Provisioning services	129
3.5.1	Architecture overview	131
3.5.2	Provisioning architecture approach	134
3.6	Conclusion	134
	Chapter 4. Deploying Tivoli Federated Identity Manager	135
4.1	Federated SSO architecture patterns	136
4.1.1	Architecture approach	136
4.1.2	Base pattern	139
4.1.3	Plug-in pattern	142
4.1.4	Lightweight Access Manager for e-business pattern	143
4.1.5	Highly available architecture patterns	147
4.1.6	Multiple data center patterns	149
4.2	Federated Web services architecture patterns	151
4.2.1	Architecture approach	152
4.2.2	Point-to-point pattern	154
4.2.3	XML gateway pattern	155
4.3	Integrating applications into an F-SSO environment	158
4.3.1	Attribute flow between providers	158
4.3.2	User-controlled federated life cycle management	161

4.3.3 Customized user-managed federation management	161
4.4 Customizing F-SSO.	163
4.4.1 Customizing page templates	163
4.4.2 Customizing Access Manager for e-business page templates	163
4.4.3 Storing aliases.	164
4.5 Solution design considerations	164
4.5.1 Exchanging metadata with your partners	164
4.5.2 Availability of IBM Access Manager for e-business policy server	165
4.5.3 Key management	166
4.5.4 Session timeout.	166
4.5.5 Application logout	167
4.6 Conclusion.	168
Chapter 5. Integrating with IBM identity management offerings	171
5.1 IBM Tivoli Access Manager for e-business	172
5.1.1 Identity provider integration.	172
5.1.2 Service provider integration	173
5.2 IBM Tivoli Identity Manager	174
5.2.1 Identity provider integration.	175
5.2.2 Service provider integration	175
5.3 IBM Tivoli Directory Integrator.	176
5.3.1 Identity provider integration.	176
5.3.2 Service provider integration	176
5.4 IBM Tivoli Directory Server	177
5.4.1 Identity provider integration.	177
5.4.2 Service provider integration	178
5.5 IBM WebSphere Application Server	178
5.5.1 Integrated Solutions Console (ISC).	179
Part 2. Customer environment	181
Chapter 6. Overview	183
6.1 Use case 1 - SAML/JITP.	186
6.2 Use case 2 - WS-Federation	186
6.3 Use case 3 - Liberty	187
6.4 Use case 4 - Web services security management	190
6.5 Conclusions.	192
Chapter 7. Use case 1 - SAML/JITP	193
7.1 Scenario details.	194
7.1.1 Contract.	194
7.1.2 User experience	197
7.2 Functionality	201
7.2.1 Single sign-on - SPNEGO.	201

7.2.2 Single sign-on - SAML/JITP	201
7.3 Partners involved.	202
7.3.1 BigCorp	202
7.3.2 RBTravel	202
7.4 Interaction description	202
7.4.1 High-level Interaction overview	202
7.4.2 Single sign-on from Windows workstation (SPNEGO)	203
7.4.3 Single sign-on from BigCorp to RBTravel (SAML/JITP)	204
7.5 Configuration data	207
7.5.1 IdP-related configuration data	208
7.5.2 SP-related configuration data at RBTravel	212
7.6 Assumptions/implementation notes.	215
Chapter 8. Use case 2 - WS-Federation.	219
8.1 Scenario details	220
8.2 Contract.	220
8.3 User experience	223
8.3.1 Single sign-on user experience.	223
8.3.2 Sign-off user experience	225
8.4 Functionality	227
8.4.1 Single sign-on - WS-Federation	227
8.5 Partners involved.	227
8.5.1 BigCorp	228
8.5.2 RBTelco	228
8.6 Interaction description	228
8.7 Configuration data	233
8.7.1 Identity provider configuration at BigCorp	234
8.7.2 Service provider configuration at RBTelco	239
8.8 Assumptions/implementation notes.	242
8.8.1 Understanding the many-to-one user identity mapping	242
Chapter 9. Use case 3 - Liberty	245
9.1 Scenario details.	246
9.1.1 Contract.	246
9.1.2 User experience	247
9.2 Functionality	254
9.3 Partners involved.	254
9.3.1 RBTelco	254
9.3.2 RBTickets	254
9.3.3 RBBanking	255
9.4 Interaction description	255
9.4.1 Liberty account federation.	255
9.4.2 Single sign-on to partners (Liberty)	261

9.4.3	Single sign-off	262
9.5	Configuration data	265
9.5.1	Identity provider configuration at RBTelco	265
9.5.2	RBTickets service provider configuration data	273
9.5.3	RBBanking service provider configuration data	278
9.6	Assumptions/implementation notes	284
9.6.1	InfoService integration	284
9.6.2	Page customizations	286
Chapter 10.	Use case 4 - Web services security management	291
10.1	Scenario details	293
10.1.1	Contract	294
10.1.2	User experience	296
10.2	Functionality	301
10.2.1	Web services security management at RBTelco	302
10.2.2	Web services security management at RBStocks	302
10.3	Partners involved	302
10.3.1	RBTelco	303
10.3.2	RBStocks	303
10.4	Interaction description	303
10.4.1	Web services security management Token Generator with Access Manager binary security token callback handler	305
10.4.2	Web services security management Token Consumer with Access Manager Credential login module	307
10.4.3	Web services security management Token Generator with Web services security management Callback handler	307
10.4.4	Web services security management Token Consumer with SAML Assertion login module	315
10.5	Configuration data	319
10.5.1	Overall architecture and prerequisites	319
10.5.2	RBTelco configuration	319
10.5.3	Outbound Web services gateway configuration	336
10.5.4	RBStocks configuration	348
10.6	Troubleshooting	357
10.6.1	Using the logs for Web services security management	357
10.6.2	Using the logs for the Secure Token Service	358
10.6.3	Using the WebSphere logs	358
10.6.4	Using TCPMON	359
Part 3.	Appendixes	361
	Appendix A. Configuring Access Manager WebSEAL and Web plug-in	363
	Introduction	364
	Identity provider integration	365

Configuring WebSEAL as an identity provider	366
Updating WebSEAL configuration file	366
Configuring a junction to Tivoli Federated Identity Manager	367
Configuring extended attributes for credentials in WebSEAL	367
Configuring Web plug-ins as an identity provider	368
Updating Web plug-in configuration file	368
Configuring extended attributes for credentials in Web plug-ins	369
Service provider integration	369
External Authentication Interface	370
Trigger URIs	370
EAI headers	370
External Authentication Interface example	371
EAI header variables reference	373
Configuring WebSEAL as a service provider	374
Updating WebSEAL configuration file	375
Configuring a junction to Tivoli Federated Identity Manager	376
Access Manager policy for trigger URLs for EAI	376
Sending extended attributes as HTTP headers with WebSEAL	377
Configuring Web plug-ins as a service provider	377
Updating Web plug-in configuration file	378
Access Manager policy for trigger URLs	380
Sending extended attributes as HTTP headers with Web plug-ins	380
Appendix B. Identity mapping rules	381
Authoring identity mapping rules	382
STSEntityUser schema	383
Mapping between STSEntityUser and native tokens	384
Tivoli Access Manager credential	385
SAML 1.0 token	389
SAML 1.1 token	392
Liberty 1.1 token	394
Liberty 1.2 token	395
UsernameToken token	397
Calling Java code from mapping rules	399
Learning how to call Java from XSL	400
Distributing Java code	400
Developer tricks for mapping rules	400
Working with Access Manager credentials	400
Testing XSL rules	401
Scenario mapping rules	403
Use case 1 mapping rules	403
BigCorp mapping for use case 1	404
RBTravel mapping for use case 1	405

Use case 2 mapping rules	410
BigCorp mapping for use case 2	410
RBTelco mapping for use case 2	412
Use case 3 mapping rules	414
RBTelco mapping for use case 3	414
RBBanking mapping for use case 3	415
RBTickets mapping for use case 3	416
Use case 4 mapping rules	418
RBTelco mapping for use case 4	418
RBStocks mapping for use case 4	420
Appendix C. Keys and certificates	425
Keys and certificates	426
Required keys	426
Keystore layout.	428
Keystores for BigCorp	429
Keystores for RBTravel	429
Keystores for RBTelco	430
Keystores for RBBanking	430
Keystores for RBTickets	431
Keystores for RBStocks	431
Importing keys	432
Appendix D. WS-Security deployment descriptors	437
Web services client at RBTelco	438
RBTelco client extension configuration	438
RBTelco client binding configuration	439
Web services gateway at RBTelco.	440
RBTelco WSGW server configuration	440
RBTelco WSGW server extension configuration	440
RBTelco WSGW server binding configuration	441
RBTelco WSGW client configuration	441
RBTelco WSGW client extension configuration	442
RBTelco WSGW client binding configuration	443
Web services server RBStocks	448
RBStocks server extension configuration.	448
RBStocks server binding configuration	450
Glossary	455
Related publications	469
IBM Federated Identity Manager manuals	469
IBM Redbooks	469
Other publications	469

Online resources	470
How to get IBM Redbooks	472
Help from IBM	472
Index	473

Archived

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law. INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

CICS®

DB2®

developerWorks®

e-business on demand™


ibm.com®

IBM®

IMS™

RACF®

Rational®

Redbooks (logo) ™

Redbooks™

Tivoli®

WebSphere®

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM Redbook discusses the federated identity management (FIM) architecture and the integration with Web services security standards and IBM Tivoli® Security Solutions. In a federated environment, a user can log on through his identity provider in order to conduct transactions or easily access resources in external domains. Partners in a federated identity management environment depend on each other to authenticate their respective users and vouch for their access to services. Federated identity standards, like those being produced by the Liberty Alliance or the Web services security specifications, form an encapsulation layer over local identity and security environments of different domains. This encapsulation layer provides the ingredients for interoperability between disparate security systems inside and across domains, thus enabling federation.

The IBM Tivoli Federated Identity Manager solution extends identity management for both the identity provider and service provider infrastructure. Tivoli Federated Identity Manager solution builds on the current Tivoli identity and security offerings.

Note: *Federated identity management (FIM)* is a standard term that is widely used in the IT industry. The *Tivoli Federated Identity Manager* is the corresponding IBM product to implement a federated identity management solution.

Part 1, “Architecture and design” on page 1, discusses architecting a federated identity management solution between trusted business partners. The related chapters describe the different standards in the federated identity management context, and architecture options for deploying Tivoli Federated Identity Manager. Additionally, approaches are shown for integrating Tivoli Federated Identity Manager with other middleware and customer applications. Furthermore, the high-level components and new concepts for the design of a federated identity management solution are introduced.

Part 2, “Customer environment” on page 181, considers a scenario that involves several hypothetical corporations, and it shows how they might be able to take advantage of identity federation to improve customer experiences, reduce costs, and improve overall security. This scenario, for example, involves two large corporations with internal employee portals. The employees of these corporations authenticate to their corporate portals and are offered access to services provided by other companies without having to re-authenticate.

Part 3, “Appendixes” on page 361, gives a detailed description of various federation configuration subjects that are common to the applications of the federation scenario introduced in Part 2, “Customer environment” on page 181.

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

The team that wrote this redbook

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



The team that wrote this book is shown in the picture above. They are, from top left to bottom right: Werner, Axel, Mark, Johan, Ray, Heinz Peter, and Shane.

Axel Buecker is a Certified Consulting Software IT Specialist at the International Technical Support Organization, 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 19 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.

Werner Filip is a professor of the department for Computer Science and Engineering at the University of Applied Sciences Frankfurt am Main, Germany and a Consultant in IT Security. His primary research interests are Systems and Network Management and Applied Security. Prior to joining University of Applied Sciences Frankfurt he worked for 25 years for IBM in various positions, during his last 10 years with IBM as a Consultant in Systems and Network Management at former IBM's European Networking Center, Germany. He received a Diploma in Mathematics, and a Doctorate in Computer Science from the Technical University Darmstadt, Germany.

Heinz Peter Hippenstiel is a Security Architect and Specialist for IBM Software Services Tivoli in Germany. He joined IBM in 2000 and has a nine-year history in security and systems management solutions. In 1999 he co-authored the IBM Redbook *Guarding the Gates Using the IBM eNetwork Firewall V3.3 for Windows@ NT*. Heinz Peter is an IBM Tivoli Certified Deployment Professional and an IBM Associated IT Architect. He received a certificate in information technology from the Akademie für Datenverarbeitung in Böblingen, Germany.

Mark Hollin is a Consulting Manager in the Corporate Information Technology division of Prudential Financial in Roseland, New Jersey. He has over 20 years of experience in information technology. The last 12 of those years have been focused on computer and information security. He holds a BA in Computer Science from La Salle University in Philadelphia, Pennsylvania. His areas of expertise include Java™ programming and security systems integration.

Ray Neucom is a Consulting IT Specialist working in the Americas Product Introduction and Exploration group of IBM S&D. He has 25 years of IT experience in applications development, systems integration, and Web application security. He has been involved with IBM Tivoli Federated Identity Federation since the beginning of the Early Support Program and has represented IBM with Tivoli Federated Identity Manager at several conformance testing events. He holds a Bachelors of Science, a Masters of Scientific Studies (both majored in Computer Science), and a graduate Diploma in Business Administration.

Shane Weeden is a Senior Software Engineer with the IBM Federated Identity Manager development team. He has worked in IT security for 13 years, and has spent the last seven years working with Tivoli Security products. Shane has been with the Federated Identity Manager development team since its conception, and now divides his time between customer-focused engagements and core product

development activities. He holds a Bachelors of Information Technology from the University of Queensland in Australia.

Johan Westman is a Client IT Architect, working in the IBM Nordic Industrial Sector in Sweden. He has worked for 11 years for IBM and is a Certified IT Specialist. He has many years of experience in working with large e-business infrastructures and service provider delivery environments with both telecom and industrial customers. In 1998 he co-authored the IBM Redbook *Understanding LDAP - Design and Implementation*, SG24-4986. He holds a Masters of Science degree in Engineering Physics from Uppsala University in Sweden, where he specialized in scientific computing.

Heather Hinton is a Senior Security Architect in Austin, Texas. She has 12 years of experience in computer and information security. She has a Ph.D. in electrical and computer engineering from the University of Toronto. Her areas of expertise include Federated Identity Management, access control, composition of policy, wireless, network, and systems security.

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

Julie Czubik

International Technical Support Organization, Poughkeepsie Center

Patrick Wardrop, Rahul Mishra, Brian Eaton, Glen Gooding, Matthew Duggan, Sean McDonald

IBM US

Gavin Bray

IBM Australia

Special acknowledgements: Many portions in this book have greatly benefited from material provided by the development architecture team and the Product Introduction Center in the US and the UK. Special thanks go to Heather Hinton, Venkat Raghavan, Jon Harry, and Avery Salmon.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll 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 Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM® Corporation, International Technical Support Organization
Dept. JN9B Building 905
11501 Burnet Road
Austin, Texas 78758-3493

Archived



Part 1

Architecture and design

This part gives an overview of the capabilities of a general federated identity management solution. These capabilities are treated as individual logical functions that may be leveraged in a FIM solution. Additionally, the high-level components and new concepts for the design of a federated identity management solution using IBM software technology are introduced.

This part provides you with an understanding of the high-level logical services architecture for IBM Tivoli Federated Identity Management, and a more detailed look into federated single sign-on (F-SSO), Web services security management, and provisioning solutions.

Finally, architecture options for deploying Tivoli Federated Identity Manager, approaches for integrating Tivoli Federated Identity Manager with other middleware and customer applications, and several important issues relating to deploying Tivoli Federated Identity Manager in a production environment are described.

Archived



Business context for identity federation

This chapter discusses the business environment and the IT environment in the context of identity federation. The business environment, driven by an increasing ability to adapt, will in larger extent be highly collaborative, involving multiple parties exchanging transactions as part of new horizontal business processes. To support the required business flexibility IT must evolve to support these new requirements.

1.1 Federated identity

Federated identity technology is used for creating a globally interoperable online business identity, driving relationships or affinity driven business models between companies. The concept is nothing new, as 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.

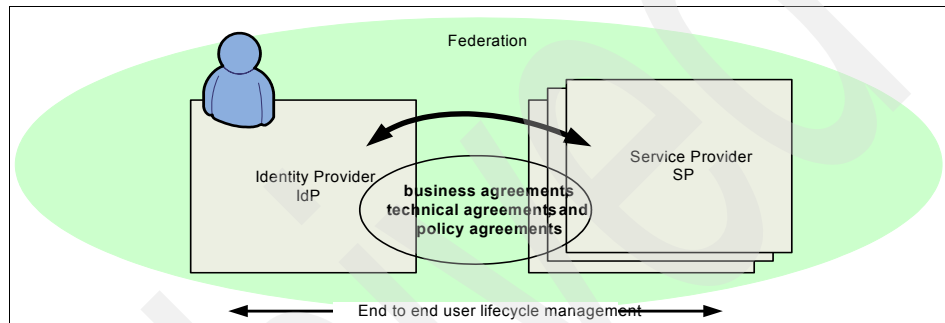


Figure 1-1 Federated identity management

Federated identity management is based on the 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. The simplification of the administration and the life cycle management in a federation leads to the following value proposition:

- ▶ Identity management costs can be lowered because companies are no longer in the business of managing *users* or *identities* that are not under their control, including the delegate administrator identities currently managed by many first-generation federation attempts. Businesses need to manage access to data but do not have to manage accounts and user account data.
- ▶ User experience can be improved because users can navigate easily between Web sites while maintaining a global login identity.
- ▶ Inter-enterprise application integration within federations benefit from the end to end security and trust capabilities.

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.

1.2 Business environment

Today, businesses are continuing the on demand evolution by developing the business models needed to create value and demand for their new products and services. In this evolution, companies are typically working toward becoming a company that with key business partners, suppliers, and customers that can respond with flexibility and speed to any customer demand, market opportunity, or external threat (IBM definition of an on demand business). Such businesses (on demand businesses) have the following four key attributes:

- Responsive** Able to respond to dynamic, unpredictable changes in demand, supply, pricing, labor, competition, capital markets, and the needs of its customers, business partners, suppliers, and employees
- Variable** Able to adapt processes and cost structures to reduce risk while maintaining high productivity and financial predictability
- Focused** Able to concentrate on its core competencies and differentiating, meeting the needs of all of its constituents
- Resilient** Able to manage changes and external threats while consistently meeting the needs of all its constituents

In the process of becoming an on demand business, businesses will go through many changes. The path of deconstruction and re-aggregation of businesses open up for new possibilities and challenges.

1.2.1 Deconstruction of the enterprise

Enterprises are de-constructing and reorganizing into extended value-nets of partners, suppliers, customers, and competitors to increase productivity and flexibility—shedding non-core processes to focus on strategic, core processes, see Figure 1-2 on page 6. This deconstruction is being accelerated by the evolution and adoption of open standards and services oriented architectures—the deconstruction is increasingly occurring at a more global scale, at an increased rate and at a more granular level.

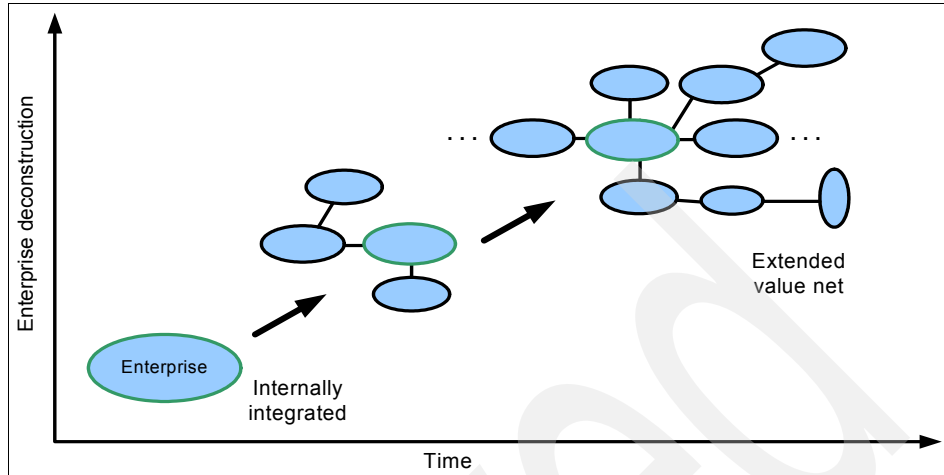


Figure 1-2 Enterprise deconstruction

The main drivers behind this change are an increasing pervasive nature of technology, open standards, globalization, and the increasing fusion of business models and IT capability.

Increased collaboration brings greater business rewards, but also poses greater business risks, and will require fundamentally new business decision and security models. As businesses evolve along the line of deconstruction of their business the demand to flexibly assemble and reconfigure business systems as business needs evolve—they re-aggregate.

1.2.2 Enterprise re-aggregation

As some components or services are moved out of the business the possibility to re-aggregate them to support dynamic and changing business models increase. The market is increasingly adopting to a strategy along which to do this. The concept of this kind of re-aggregation or integration is well reflected in the Service Oriented Architecture (SOA) strategy.

SOA is an approach to defining integration architectures based on the concept of a service. The business and infrastructure functions that are required to make an effective on demand environment are provided as services. These services are the building blocks of the system. For more on SOA there are many Redbooks on the topic, for example: *Patterns: SOA with an Enterprise Service Bus in WebSphere® Application Server V6* , SG24-6494.

Services can be invoked independently by either external or internal service requesters to process simple functions, or can work together by choreographic implementations to quickly devise new functionality to existing processes.

SOA may use Web services as a set of flexible and interoperable standards for distributed systems. There is a strong complimentary nature between SOA and Web services. Software components and Web services are well positioned to provide the flexibility, well-defined business-level concepts, and the mechanism to achieve assembly and re-configuration of the business systems required.

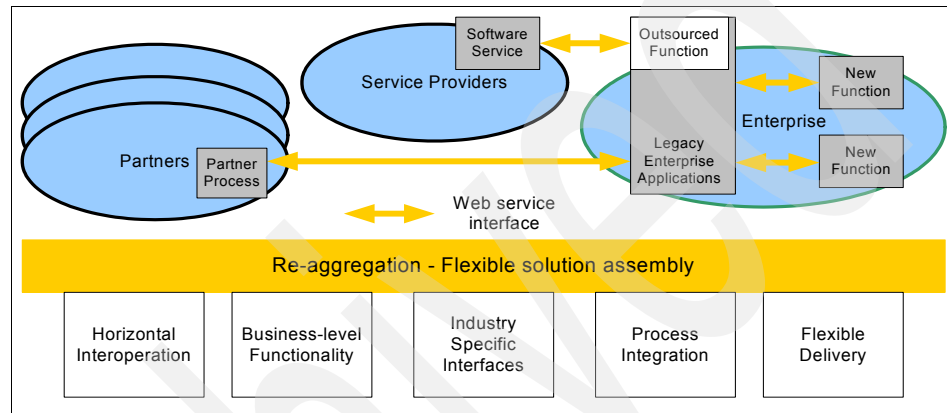


Figure 1-3 Re-aggregation for solution assembly

In Figure 1-3 is an example of a solution assembly of services that have been externalized as a result of deconstruction of an business. The re-aggregation is accomplished using standards based Web services. This is elaborated on in 1.3.4, “Inter-enterprise application to application integration” on page 25.

It is important to remember that inter-enterprise application integration is not always the most efficient way to address the implementation of a cross function or business process. The business process, as is common today, is fulfilled by integrating the user experience across multiple user interfaces, represented by Web servers presenting the application. Today, this is just navigating the Web to a large extent, but as portals become more pervasive and their content is spread across the Internet, the user integration is solved by federated single sign-on. This is integrating “on the glass” or at the user interface. A combination of both inter-enterprise integration and user interface integration is the major part of what federated identity management is about.

1.2.3 High-level example of a re-aggregated business

In a world where more and more services will be technology enabled, including areas that require the exchange of extremely private and sensitive information,

the current reactive approaches to resource (service) deployment does not satisfy the requirements for the real-time, fluidly aggregating services to optimally address shifting market conditions.

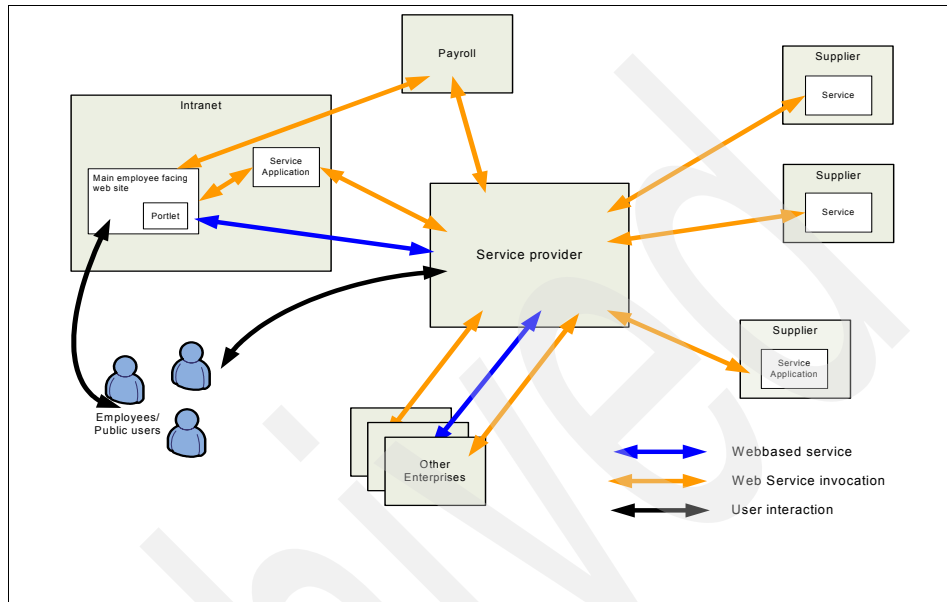


Figure 1-4 Re-aggregation of businesses into complex value webs multiplies concerns

Figure 1-4 illustrates some of the integration points that must be addressed as part of the re-aggregation of services to support new or existing business processes. A company (intranet in Figure 1-4) *outsources* the administration of its telecom and related services (to the service provider) and also its payroll services (to Payroll, shown in Figure 1-4). The service provider in turn has similar relationships with other customers, and with its own suppliers. Note that the service provider also outsources its payroll activities to the payroll entity.

These kinds of relationships are in place today at many businesses, but it is a difficult process to implement, customize, maintain. For example, a business process that must be shared across organizational domains presents several follow-on challenges such as workflow across company boundaries. The administration provider must aggregate services provided by its suppliers into a coherent set of unified services that it in turn supplies to its customers.

Furthermore, the supplier must ensure and guarantee that information is secure, segmented, and private among its customers. The suppliers must communicate with one another on behalf of the employees, maintaining privacy. The supplier may need to know its users, which may be numerous.

Understanding and control of complex dynamics and collective behavior will become increasingly important to avoid system instability and set the stage for both global and local optimization. A fundamentally new approach needs to be implemented to provide a secure foundation for the transformation to on demand that includes federation and containment.

So, the interactions required to fulfill the new business processes requirements will be a mix of application to application and user interactions, requiring the full set of federated identity management capabilities to handle the challenges of identity, trust and security.

1.2.4 Business models for federated identity

In this section, some possible examples reflecting the challenges within the changing business and IT environment that are relevant to the federated identity management area will be studied in more detail.

In an attempt to make the examples relevant for as many businesses, enterprises and authorities, examples are taken from five different sectors.

Financial	The financial sector is represented by banking, financial markets, and insurance.
Communications	The communications sector is represented by energy and utilities, media and entertainment, and telecommunications.
Distribution	The distribution sector is represented by consumer products, retail, travel and transportation, and wholesale distribution and services.
Industrial	The industrial sector is represented by aerospace and defense, automotive, chemical and petroleum, and electronics.
Public	The public sector is represented by education, government and healthcare, and life sciences.

First lets just take a look at where the different sectors are in their deconstruction phase.

Many Industries are Undergoing Deconstruction into Extended Value Nets

Industries Are in Different Phases of Deconstruction

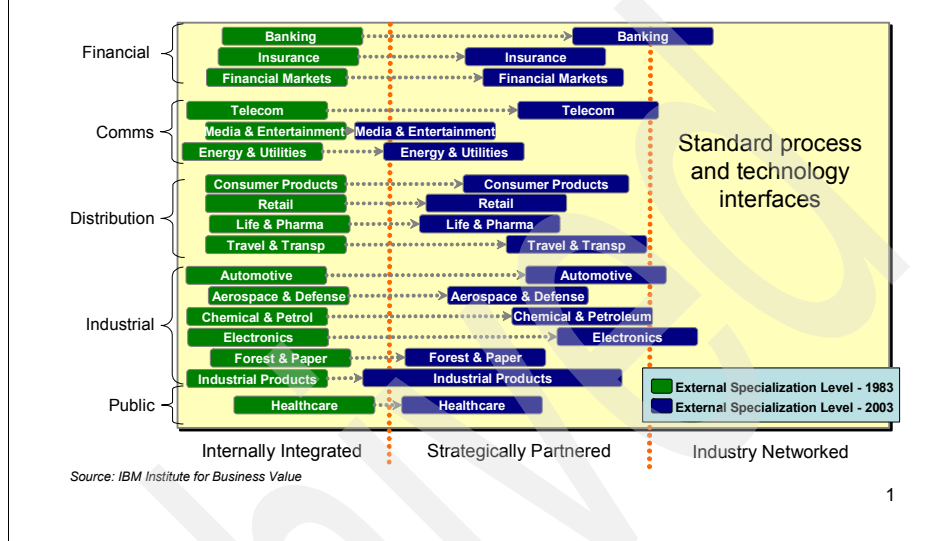


Figure 1-5 Industries are in different phases of deconstruction - Source: IBM Institute for Business Value

According to a study by the IBM Institute for Business Value, today most industries have moved into the *strategically* partnered category. In this category, the interfaces shared between businesses are still one-to-one and not re-usable. As time passes, deconstruction will move into “industry” networked (also referred to as extended value net), where interfaces will be governed by standards with a smooth transition of process, technology, and ultimately people. This move, as mentioned earlier, is one of the major drivers for looking at federated identity management.

Since sector industries are at different phases of their evolution of deconstruction the urgency to address federated identity management varies. Telecom, banking, automotive, travel and transportation, and electronics being the furthest ahead, they will be focus points for the examples and to some extent the use cases chosen in Part II of this book.

The chosen examples are:

1. Mergers and acquisitions
2. Cross-business unit collaboration

3. Growth of customer base
4. Outsourced services
5. Service provider automation
6. Portal based integration
7. Government collaboration
8. Corporate governance

In Table 1-1 there is a table show high relevance of the different examples in the different sectors. Table 1-1 shows how relevant these examples are for the different sectors.

Table 1-1 Examples relevance to different sectors

Sector \ Example no.	1	2	3	4	5	6	7	8
Communications - Telecom	X	X	X	X	X			X
Distribution	X	X		X				X
Financial - Banking	X	X		X	X			X
Industrial	X	X		X		X		X
Public - Government		X					X	

For a more detailed discussion of federated identity managements use in different scenarios, look in Chapter 6, “Overview” on page 183, where there are a few scenarios in which some of the business examples are represented.

1. Mergers and acquisitions

In this scenario a company is implementing a growth strategy using mergers and acquisitions. The indicator of the success of the merger or the acquisition is predicated on how quickly the companies can knit together their IT infrastructures to target and cross-market to the new customer base. Identity management is one of the most complex activities in such mergers. Rather than having to forklift all of the acquired users in the various systems, an integration strategy based on identity federation can simplify the user experience. The combined users of the merged customers can have access to the shared assets of the merged companies without impacting user experience, customer care, or the quality of support. Federating the identities between the merged companies provides a quick and seamless way to integrate the customers of the two companies to drive merged growth scenarios.

2. Collaboration between autonomous cross-business units

Many large companies have independent business units that want to directly maintain ownership and relationship with their users. This may be due to organizational structure, or to political, competitive, or regulatory reasons. A

large global manufacturing company may be organized as independent companies with regional management consolidated in the Americas, Europe, Africa, Middle East and Asia. However, these business units may also need to have their users (employees) needing access to cross-business unit resources. For example, employees in Asia need access to ordering and parts information in other regions. Federated identity management enables business units to retain autonomy and control of their users, yet have a flexible way to federate data to cross-business unit resources.

3. Customer acquisition strategy via partnerships

A company whose growth strategy is based on acquiring new customers needs to either obtain these customers outright or have partnerships with other companies to target their customers. A financial services provider may form a partnership with a mobile wireless provider (with millions of subscribers) to deliver paperless e-billing to these customers. The incentive for the mobile wireless provider in this partnership is to reduce their non-core expenses by outsourcing billing functions to the financial provider. In return the mobile provider would offer a 5 percent discount for customers subscribing to the new e-billing service, thereby offering an incentive for the customers to sign up for e-billing. Through this partnership, the financial services company now has acquired a million new customers to which it can target its e-billing service. Federated identity management will enable the financial service company to access large pools of customers having a well-established identity.

4. Employee access to outsourced provider services

Employee self-service is a major initiative for many companies looking to reduce user provisioning or user care costs. Most organizations outsource non-critical competencies to third-party providers. The services that are being outsourced include human resources, employee savings plan, healthcare, payroll, travel and procurement services. Using the corporate intranet portal to connect the employees directly with these external service providers enables the organization only the administration of these outsourced services. Organizations outsource these services to reduce these service administration costs. However, the inability to directly connect the employee to these service providers means that the organization is now required to support and maintain staging systems (help desk) for employee enrollment to savings plans (401K or super-annuation), healthcare or payroll. Employers spend significant amount of plan administration costs in employee 401K administration, employee stock options, employee healthcare and travel. These services are typically outsourced to various outside providers. However, the company still ends up manually administering these plans or having to staff customer care personnel for employee management.

Federated identity management provides a compelling value proposition in this scenario by enabling employees to access and manage their data on the various

third-party service provider Web sites by simply signing on to the employee portal. Access through an existing portal can simplify the user experience, and enables the user to interact with various employee provider sites without requiring additional enrollment, registration or authentication to these business partner sites. The employer in turn can lower their employee support and plan administration costs by enabling employees to interact directly with the various providers.

5. Service provider automation with B2B clients

A larger service provider managing retirement accounts for employees, pension plans, employee stock options, or healthcare for their institutional clients may incur tremendous cost of user life cycle management of its clients' employees. These costs can result from having to register and provision online accounts for client employees, manage passwords, and staff a help desk for dealing with user access problems resulting from forgotten user names, credentials or passwords. Assume an average password reset call costs \$20, and that there exists a service provider who manages 100 Fortune clients, each of whom on average have 10,000 employees. Even if only a quarter of these employees forget their password just once a year, this would represent a \$5 million annual cost in managing user accounts and passwords. The service provider is heavily motivated to move to a federated model where the service provider leverages the employee's corporate portal authentication to provide access to their services. In this model, the employer (client) is responsible for managing its users and passwords (the client does not face additional costs, because they already have to manage these users and passwords), and the service provider offloads the cost of user administration to its clients. This approach also benefits the employee tremendously, as the employee does not have to register or remember a separate sign on and password to manage their 401K or healthcare.

6. Portal-based integration of software-as-services

A new generation of Internet-based providers now delivers software-as-services to companies or corporations. Examples of these software-as-services are providers like WebEX, Salesforce.com, Siebel CRM On Demand, Travelocity.com, and so on. These services enable companies and small businesses to access Internet hosted services without having to undertake the IT infrastructure cost of managing these services locally. Federated identity plays a critical role in this system by enabling employees of the companies to access various software-based services using their employee identity sign on. As more and more non-core business services are being outsourced or offloaded with providers, federated identity management fulfills the role of an identity integration technology that enables the user to seamlessly access third-party services that may be locally hosted, remote-hosted or accessed by a software-as-services provider.

7. Government collaboration

Governments have high demands on efficiency and ability to collaborate. Many processes will span multiple governments, institutions, authorities or agencies in many regions, who will need to share data, but due to political, organizational or other challenges will not be able to consolidate or internally integrate. All these entities may also need to have their users (employees, citizens) have access to cross-governmental entities resources. For example, authorities in one European country may need to find relevant information about a person in an other countries data source, but each country would not like to manage all other countries authorities users (to maintain traceability on citizens person data). Federated identity enables authorities to retain autonomy and control of their users, yet have a flexible way to federate data to cross-governmental entities resources.

8. Improved corporate governance

Corporate governance and complying with various regulations may be major initiatives at companies. Compliance with Sarbanes-Oxley (SoX), Basel II, and HIPAA are at the forefront of the concerns of many executives.

One of the key impediments to passing an audit and achieving compliance is lack of accountability for granting user rights and permissions to access business systems. A primary reason for failing an audit is the inability to account for access rights granted to business partner users.

Federated identity can ease some of the burden associated with the following compliance pain points:

- ▶ Organizations cannot account for access rights granted in their internal systems for third-party users; there is no proof of whether a third-party user actually exists or even needs access.
- ▶ No accountability on why a third-party user was granted access in the first place; failure to demonstrate and document the business reason for the request and which company officer approved the request.
- ▶ No procedures in place to delete entitlements or purge user access rights belonging to third parties and their users. This results in users accumulating access rights far beyond what they were originally authorized.
- ▶ No procedures in place to de-provision user accounts when users turn over. This issue is magnified when dealing with third parties when the company does not control the third-party user and no process typically exists by which third-party companies will notify of user turnover.
- ▶ No way to re-certify third-party user access. Does this third-party user still need access beyond three months or six months? Why do they still need access?

- ▶ No way to audit request for third-party access. Most companies are not able to audit third-party user access in a centralized fashion because there is no one single tool that is being used to grant third-party access.

In today's model where the company takes on the management burden of third-party user administration and provisioning, these audit issues are magnified when these third-party users turn over and this identity is not propagated to the company for de-provisioning. There is no way for the company to know that a business partner employee is no longer employed. Federated identity improves compliance by offloading user administration costs to business partners. Since the company does not own the user account management accountability, approvals and re-certification are now offloaded to business partners. The company relies upon its business partners to authenticate and issue credentials that vouch for its users. The burden of proof now belongs to the business partner for vouching for its own access rights. Federated identity provides a strategic alternative for companies to simplify their administration and improve governance by offloading third-party user management to their clients.

1.2.5 The relationship - Trust and assurance

A federated business model mandates *a foundation of trust*. In a federated model an organization is willing to provide access to an identity that is not vetted by the organization's own internal security processes. Instead the organization is trusting an identity asserted by a third party, a model that introduces risk and uncertainty in the overall confidence of the business transaction.

An organization will not engage in a federated business model if they do not have the visibility into their business partners' identity and access management systems and processes. An organization needs to evaluate the risk of conducting business with business partners and needs to assess their business partner's processes and vetting procedures for 1) business partner identity proofing, 2) business partner accreditation, and 3) business partner reputation evaluation. These procedures provide the visibility and the qualitative assessment of how third-party identity can be parlayed into business decisions about access control and the rules of engagement around trust that the organization is willing to enter with the business partner company.

Business partner identity proofing is the process of verifying the physical identity of a prospective federation business partner both before entering into an online business relationship with that business partner and when engaged in runtime transactions with the business partner. Part of the business partner identity proofing process involves verification of the physical identity of the business—but who is the business?

- ▶ Is there a legitimate business with the stated name?
- ▶ Is this the party making the request?

- ▶ Is the specific employee making the request authorized?

Once the physical identity has been verified, some form of online token is issued to the business partner and then bound to the physical identity of the business.

Various forms of business partner identity verification techniques and processes can be used, including:

- ▶ Self-assertion
- ▶ Leverage of an existing relationship
- ▶ Confirmation of electronic or postal address
- ▶ Credit agency, business bureau ratings
- ▶ As the name suggests, identity verification

Business partner accreditation addresses the question what do we know about the company? And more specifically, what do we expect of this company? Accreditation is based on a well-defined policy that defines the criteria that a company must satisfy. A company that wishes to enter into a federation may publish a policy that defines the criteria that prospective business partners must match; likewise, a business partner wishing to enter a federation may publish a policy that defines the criteria that IT satisfies (a policy describing its own features). Evaluating the fit of these two policies is an action that is undertaken by a trusted party specializing in business accreditation.

Examples of the types of characteristics that are evaluated as part of the accreditation process include:

- ▶ Is the company credit worthy?
- ▶ Is the company considered to be a reputable business?
- ▶ Is the company approved by relevant professional/trade bodies?
- ▶ Is the company part of the federation?
- ▶ Has the company authenticated and issued credentials in a standardized trustworthy fashion?

Reputation is an alternate means of knowing additional qualitative information about a business. The primary difference between a reputation service and accreditation is that reputation typically is measured on an ongoing basis using behavioral information about the business or an individual. Another difference is that reputation is typically measured by an independent entity and typically does not involve the participation of subject (business or individual being measured). The reputation service may develop an automated framework for measuring reputation based on transactional visibility. Alternatively, a more explicit feedback-based mechanism is used. The reputation service will usually assign a simple score that is derived using a well-defined procedure and is easy to understand.

Organizations face critical challenges in determining the risk/return relationship in a federated model. Business partner identity verification, accreditation and reputation are basic tenets that help companies determine their level of trust and assurance in their business partners' identity management solution.

1.3 IT environment

Driven by the changes in the business environment IT needs to adopt to support the new emerging requirements. This is to a large extent done by the concept of services oriented architecture and single sign-on. To enable this kind of dynamic integration between entities federated identity management becomes key.

In this section IT challenges in identity management are studied. The focus areas are how to deal with identities, user life cycle management, user provisioning, account management, inter-enterprise application to application integration.

1.3.1 The role of identity management

Note: The IBM Federated Identity Management white paper (Heather Hinton, et al) has been used extensively in the writing of this chapter.

Identity management has become a hot topic these days with many organizations. From business unit executives to CIO's to IT administrators, the focus is on improving the integrity of identity-driven transactions, improve efficiency, and lower IT costs. Identities pervade every aspect of e-business. Corporate IT accounts (e-mail, NOS, LDAP, UNIX®, Linux®, Windows, RACF®, Desktop), HR accounts, supply-chain accounts, healthcare, 401K, online travel, and VPN accounts are all essential accounts that need to be provisioned for a new employee or a user to do their job. Few of these identities or accounts work together, so they add substantial administrative and customer support costs and deliver poor end-user experience due to multiple sign-ons to systems and applications. With increased corporate governance and regulatory hurdles, the management of these identities and account data introduces new business compliance issues and security exposures. Taking on identity management means dealing with these privacy, compliance, legal and regulatory issues.

The cost and complexity of identity administration in today's environment is primarily due to a single reason: To provide access to a user for a service or an application means giving the user an account within the service or application-specific repository. The fundamental practice of creating and managing user accounts leads to various administration, single sign-on, and compliance issues.

User life cycle management of identities

Federated Identity Management (FIM) addresses this problem by providing a standardized way of managing the end-to-end life cycle management of identities both within and between organizations. This end-to-end user life cycle management extends a company's identity management practices and procedures to simplify identity and access administration for third-party user access and simplify user access to simplify third-party resources.

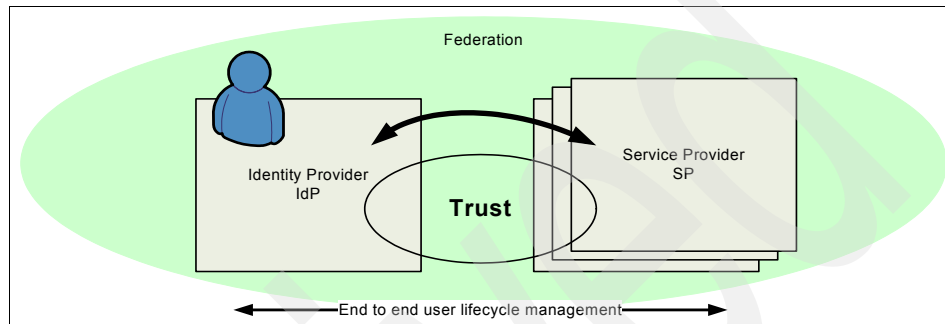


Figure 1-6 End-to-end user life cycle management

This life cycle management approach builds on a foundation of trust and incorporates standards for user identification, authentication, access control, and the exchange of identity and attribute information between services providers and service consumers. This approach helps companies to lower identity management, access management, and administration costs related to third-party user access or third-party service access.

Federated identity

At a fundamental level, the term *federated identity* has various meanings. The term identity used in a *federated context* is composed of federated attributes that can be sourced across multiple federated and authoritative data sources. There are many attributes that can represent a particular identity. The concept of identity needs to be thought of as a distributed concept where multiple attributes of an identity are federated across multiple data repositories.

To an individual user, *federated identity* means the ability to associate his various application and system identities with one another. To a business, federated identity provides a standardized means for allowing businesses to directly provide services for trusted third-party users or users that they do not directly manage. It refers to the ability of one business to associate with one or more others in a federation, such that the identities from one business domain (or *identity provider*) are granted access to the services of another business (or *service provider*).

Partnership-based solutions

Federation enables businesses to deliver solutions that can be more functional and cost-effective, and better customer acquisition strategies via federated business models. The federated business model enables service providers to be able to federate data to large established clients, business partners, and customers that they normally would not have access to.

Federated identity management refers to the set of *business agreements*, *technical agreements*, and *policies* that enable companies to lower their overall identity management costs, improve user experience, and mitigate security risks for Web services-based interactions.

IBM has recognized that federated identity management is a technology that can help companies simplify their user administration and security administration while improving security and corporate compliance. This life cycle management approach enables company administrators and auditors to have the visibility, controls, and the workflow to engage in federated administration with their business partners.

Security characteristics

In a B2C or B2E¹ environment where consumers and employees communicate with one company as a focal point for multiple business partners, it is important to secure access to all involved parties. In B2B environments business partners and applications must also be used in a secure and reliable way.

Managing identities in this dynamic environment with many different organizations interlinked becomes problematic when using today's traditional, static models. For this reason is it necessary to organize federations in order to propagate identities across multiple organizations dynamically in a seamless management infrastructure.

In such a dynamic environment, *trust relationships* between business partners are essential. Traditionally, IT infrastructures have dealt almost exclusively with their own environments—not necessarily reflecting the needs of *interoperation* and *integration* with other parties. In an truly dynamic business environment all parties must interact seamlessly to meet the requirements of a dynamic business. Figure 1-7 on page 20 highlights a security triangle that these three elements form.

¹ B2C: Business to Consumer, B2E: Business to Employee, B2B: Business to Business

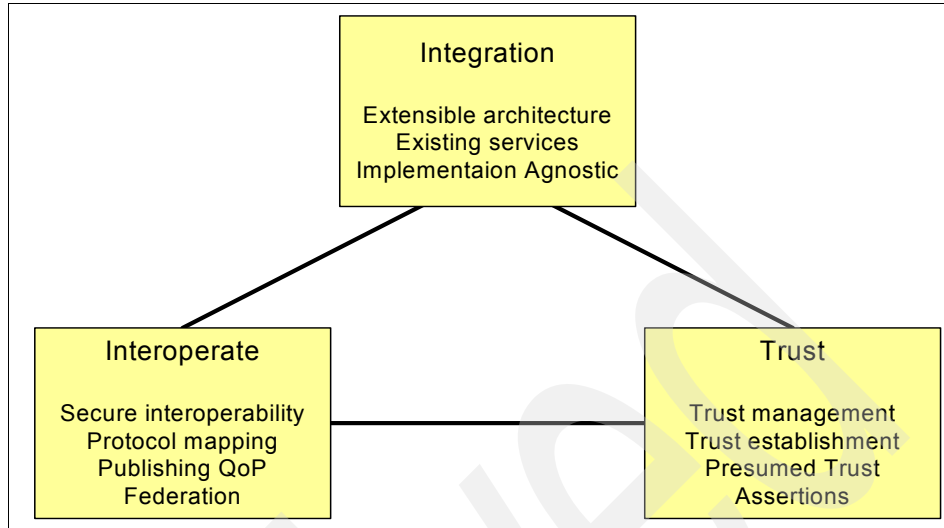


Figure 1-7 Security triangle: Trust - Interoperation - Integration

Traditional security issues, of course, still apply, but need to be expanded in many ways. In an on demand world closer convergence of IT and interlocked business require flexible architectures to reflect the needs of these virtual organizations.

Perhaps the most significant change, is the move from a static security environment to a highly dynamic environment reflecting fast changes in this world. These new security challenges span multiple organizations and are no longer bound to persons, but extend to applications and devices, as well.

New federated identity management specifications, that extend existing Web services and federation-related standards, form the basis for a solution to the new identity management issues that arise in an dynamic business environments. These solutions will be discussed in more detail throughout the remainder of this redbook.

1.3.2 Dealing with identities

A typical business that deals with at least three major clusters of identities are shown in Figure 1-8 on page 21.

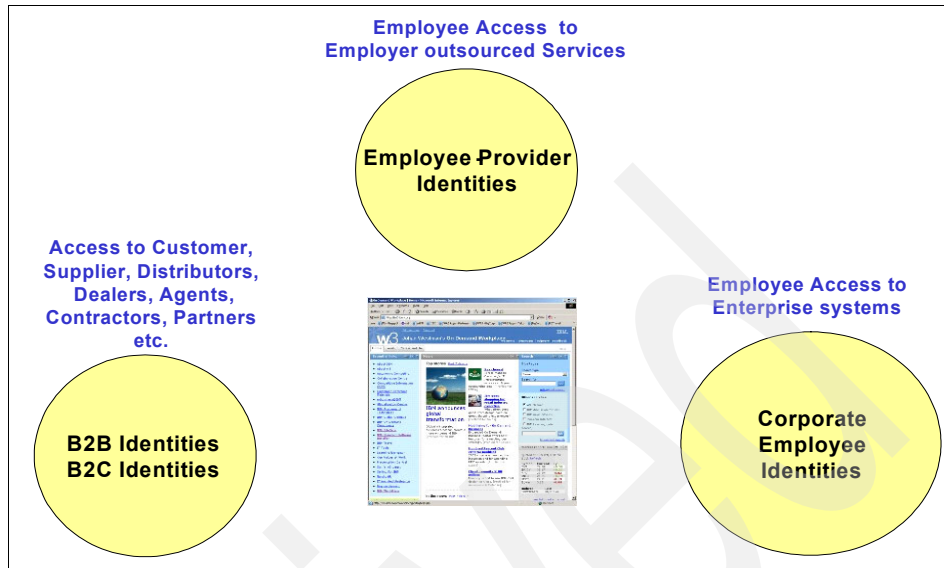


Figure 1-8 Dealing with identities - A corporate view

Attaining these goals using IT as a productivity lever has been both problematic and challenging. In the IT world seemingly simple things like managing identities or exchanging identity information within a firm's heterogeneous systems is a challenge today, not to mention trying to deliver data transparently to users from across a network of business partners and affiliates. Fundamental issues like end-to-end identity propagation are lacking today and present significant challenges to integrating identities (and identity management techniques) seamlessly into the application and middleware.

A quick survey within a typical large organization reveals many forms of identity accounts that are provisioned by the employer to employees (including employee-like users such as contractors), consultants, and contractors.

Corporate identities

A corporation typically has a number of systems and applications where their users need identities. The user needs to sign on to her workstation, possibly again to her corporate intranet, and may need to sign on again to the back-end systems. These sign-ins may need multiple identities, which need to be managed as well as the user needs to remember all of them.

- ▶ Network identities (Remote Access, VPN or Wi-Fi Accounts) to enable users to access the network
- ▶ Desktop identities to sign on to the workstation (Windows credentials)

- ▶ Corporate e-mail and white pages accounts
- ▶ Legacy accounts for mainframe accounts
- ▶ HR accounts (PeopleSoft, SAP, Oracle)
- ▶ Supply Chain/CRM accounts (SAP, Siebel, and so on)
- ▶ Identities that are managed in middleware and database solutions (Oracle accounts, WebSphere accounts, Portal accounts, and so on)

Employee to employer-outsourced provider identities

Many employee services (such as employee savings plans, retirement accounts, pension, employee stock options, healthcare, payroll, and travel services) are typically outsourced. However, employees need to register and enroll at these third-party Web sites to get a login account before they can access these services. Many small- and medium-sized businesses typically outsource many aspects of their non-core services such as customer management, payroll, and financial accounting, and so on.

- ▶ Employee benefits accounts (401K, pension, stock options, healthcare, online travel, and so on).
- ▶ Employee access to *Software-as-Services* identities. These are identities to access hosted software like WebEX, ADP, quicken.com, Salesforce.com, Siebel CRM On Demand, and so on.
- ▶ Accounts at financial service providers (IRA, 401K).
- ▶ Online banking/bill payment accounts.
- ▶ Accounts with credit card providers.

Business to consumer identities

Companies have to deal with many forms of identities to deal with suppliers, business partners, distributors, dealers, and so on. Customers need login identities to access various applications in the company portal.

- ▶ Suppliers need login accounts to access procurement systems such as SAP, and so on.
- ▶ Business Partners need accounts in various systems.
- ▶ Distributors and dealers need access to various line of business applications.

The unique element about business-to-consumer is the scale of millions of these B2C identities and accounts that need to be maintained.

1.3.3 User life cycle management

One of the biggest challenges customers face today is cost and complexity of user life cycle management. User life cycle management is also referred to as the *multiple identity account* problem, as users in most large companies have to deal with fifty plus accounts. The customer pain points today can be characterized in these facts:

- ▶ Improve and increase confidence in business transactions.
Identity is the basis of security; poor identity management means weak security.
- ▶ Lower administrative cost.
Soaring costs with account information administration and password administration, user registration, and help desk support.
- ▶ Risk, compliance, security exposure.
 - Business, legal and privacy issues with user data access (for example, Sarbanes-Oxley, HIPAA, Graham-Leach-Bliley, CA SB1386)
 - Issues with unauthorized access from users
 - Audit failures due to inactive user account exposure
 - Identity and password theft
- ▶ Poor market reach.
 - No standard mechanism to trust identities from M&A, business partners, and third parties
 - High cost of integration applications that deal with identities

The fundamental issue pervading identity management is that every time a user requests access to an application or a system, an IT administrator ends up creating an account for the user in the target system or application. A company takes on a significant cost of user administration and management when creating accounts for users.

To a great extent, these issues all involve the subject of *identity management*.

User provisioning and account management costs

The cost of provisioning users with account data is one of the more expensive and manual activities that take people, time, and a significant IT budget. While automated user provisioning tools automate (synchronize) many aspects of user provisioning, the fundamental issue still remains that a company takes on *user ownership costs* when provisioning account data. While a company may need to take on this user ownership cost for employees, this approach may not be

correct when dealing with external identities that are currently being provisioned in the internal systems.

Let us take a look at the various provisioning activities that a company is undertaking when they decide to give access by creating an account.

- ▶ Create an account in each target system for the user.
- ▶ Enrollment or registration of user in accounts.
- ▶ Establishment of access rights or credentials ensuring the privacy and integrity of account data.
- ▶ Establish initial password/PIN.
- ▶ Help desk or customer care support to handle the following:
 - Manage forgotten user name
 - Forgotten passwords
 - Managing password resets
 - Requesting new access
- ▶ Manage password synchronization.
- ▶ Manage changes to access rights as user changes roles, and to entitlements due to organizational changes.
- ▶ Eliminating access rights.
- ▶ De-provisioning accounts when user leaves the company.
- ▶ Ensuring the privacy and integrity of account data.

Every time an account is created an IT provider is buying into a set of management pain points. The key question for an IT provider becomes to decide whether he has to manage this account or is there a better way to manage access to this set of users?

There exists an opportunity for the company to reduce the cost of provisioning suppliers, business partners, consultants, brokers and third-party users. By federating user access to these third-party users, companies can effectively off load user administration costs back to the provider who has direct responsibility for managing the user.

User registration and enrollment costs

There are costs associated with registering and enrolling a new user in the systems. User registration and enrollment costs accrue from the administrative processes that need to be deployed across the IVR², Web, and sales channels. These administrative processes require evaluating of the user registration data, collecting approvals, and integrating customer care processes to handle user

² Interactive Voice Recognition

access issues. Many service providers such as managed health care providers incur significant customer care costs (for their client employees) every year during plan enrollment times. These managed healthcare or financial providers deliver services to employees of their clients. As a result, in today's business model, these providers end up with the responsibility of identity management, password management, and customer care for their client employees. Users call into the service provider support desk when they cannot remember their online user name or ID or PIN number, or are having difficulties with registration at the last minute.

This cost of user administration can be significant for most service providers and presents a recurring cost overhead. If a provider has 500 fortune clients (a client refers to a company), and each client on average has 20,000 employees whose healthcare need to be managed, the provider is now supporting and servicing 10 million accounts. A federated model where the service provider trusts its clients to provide the user information can considerably simplify user administration costs because user service costs are being handled by their clients, not the provider. In this model when an employee cannot access the healthcare enrollment page (for whatever reason, such as forgotten user ID or password, and so on) they call their local help desk for assistance. This approach greatly reduces the cost of user administration, service, and ongoing customer.

Password management costs

A significant pain point for most companies is cost of password management. Each call to the help desk results, on average, between \$20 to \$30 per call in support costs (shown by various studies).

Therefore most providers have an incentive to lower password management cost by either automating password resets or avoiding this password management problem all together. Federated identity presents an opportunity to avoid this problem altogether by enabling organizations to leverage their business partner to manage these passwords and credentials.

1.3.4 Inter-enterprise application to application integration

As mentioned in 1.2.2, "Enterprise re-aggregation" on page 6 SOA is a key strategy that the market is adopting to support the businesses drive towards becoming on demand businesses. Here we focus on the application to application integration challenges within SOA. SOA as mentioned earlier spans the own private business into new inter-enterprise interactions.

Important: While Web services provide the technology that is used for application-to-application interactions, they are not a requirement for an SOA or ESB environment. Federated identity management techniques can be used within a Web services environment, be it SOA, ESB, or based on other technologies.

The SOA strategy touches on many key elements relevant for e-business on demand:

- ▶ Interfaces are provided to wrap service endpoints to provide a system-independent architecture to promote cross-industry communication.
- ▶ SOA can provide dynamic service discovery and binding, which means that service integration can occur on demand.
- ▶ SOA provides a standard method of invoking Web services (business logic and functionality) for disparate organizations to share across network boundaries.
- ▶ Web services use open standards to allow inter-enterprise connectivity across networks and the Internet:
 - Messaging protocols (SOAP)
 - Transport protocols (including HTTP, HTTPS, JMS)
 - Security can be handled at both the transport level (HTTPS) and/or at a protocol level (WS-Security)
- ▶ WSDL allows Web services to be self-describing for a loosely coupled architecture.
- ▶ A key principle of SOA is that services should be invoked by service requesters that are oblivious to service implementation details, including location, platform, and if appropriate to the business scenario, even the identity of the service provider.
- ▶ Standards bodies, including WS-I, W3C and OASIS exist using technologists from industry leading software vendors (IBM, BEA, Oracle, Microsoft®, and so forth) to accelerate and guide open standards creation and adoption.

The Enterprise Service Bus (ESB)

A core component of realizing an on demand infrastructure enabling support of the emerging on demand business models is the Enterprise Service Bus (ESB). The ESB is to SOA as what SOA is to e-business on demand. So how does the Enterprise Service Bus address the vision of an on demand business?

The Enterprise Service Bus is emerging as a service-oriented infrastructure component that makes large-scale implementation of the SOA principles manageable in a heterogeneous world.

On demand applications are business services built from services that provide a set of capabilities that are worth advertising for use by other services. Typically, a business service relies on many other services in its implementation. Services interact via the Enterprise Service Bus, which facilitates mediated interactions. The reference architecture for integration in 2.8, “On demand integration reference architecture” on page 75, is based around SOA and an ESB.

When extending the ESB to support the inter-enterprise interactions driven by SOA, trust and security is required. If using Web services, which are assumed here, Web services security is a desired capability to allow businesses to exchange sensitive data in a secure and trusted manner. This includes secure communications across a multi-hop environment enabling application end to end security and trust.

Web services security removes the dependency on transport-level security that has been an artifact of Hypertext Transfer Protocol (HTTP)-based communications and extends it to an end to end application interaction security solution.

1.3.5 Open standards

Open standards are a key component when enabling inter-enterprise interactions especially if they are to be dynamic and loosely coupled. Just as the Web browser based user interactions have benefitted from HTML and Java based technologies, federated identity management benefits from the defined SSO protocols and Web services standards.

See more detailed description of open standards relating to federated identity management in 2.3, “FIM standards and efforts” on page 51.

F-SSO standards

Federated single sign-on (F-SSO) standards relate to how parties involved in a federation communicate with each other and how they assert identities. In the SSO standards there are also standards relating to single sign-out and account linking capabilities.

Web services

Web services have emerged as the most promising development to address cross-enterprise, cross-platform, and cross-vendor business integration issues. Web services is a family of emerging technologies that enable easy

interoperability of programmed information technology (IT) services and integration of applications into a company's broader business processes. Web services technology enables companies to describe available services and provide access to those services over standard Web protocols and communications boundaries.

Web services security specifications

In April of 2002, IBM and Microsoft published a *Web services security roadmap*. This roadmap describes a modular set of Web services specifications that allow customers to build secure Web services according to their individual needs. Several of these specifications have since been published and are described in this section. You can download the roadmap from the Web at:

<http://www.ibm.com/developerworks/library/ws-secmap>

The Web services security roadmap defines and describes a set of specifications designed to provide a security standard foundation. This foundation is based on *WS-Security*, *WS-Trust*, *WS-Policy* and *WS-Federation* specifications. These specifications provide a high-level view of all the pieces needed for security in a Web services environment. In addition, these security specifications are factored with the rest of the Web services architecture. This allows customers to easily add other critical functionalities such as reliable messaging or transactions to a Web service.

Web services provisioning specification

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.

The specification is publicly available on the IBM developerWorks® Web site:

<http://www.ibm.com/developerworks/webservices/library/ws-provis/>

1.4 Conclusion

Organizations are looking to increase productivity and efficiency in both their intra-enterprise and inter-enterprise interactions. Keys to productivity are to reduce cost, reduce friction and promote reuse. Most organizations are moving to a *services-based delivery model* or *service-oriented architecture* where business services are available through the integration of loosely coupled application platforms.

Federated identity management delivers clear and compelling business productivity by reducing the friction caused by incompatible identity management systems. Since identity is a fundamental tenet of business and since organizations have a business need to integrate their systems and applications together, federated identity offers a strategic opportunity for companies to address both issues. It provides the glue that enables organizations to network and integrate their application platforms securely using Web services. Federated identity management enables companies to securely link, join, or extend their IT infrastructures with those of their business partners rather than create and manage redundant identity and security infrastructures.

IBM has recognized that federated identity management is a user life cycle management and administration problem. This approach enables companies simplify their user administration and security administration while improving security and corporate compliance. This life cycle management approach enables company administrators and auditors to have the visibility, controls, and the workflow to engage in federated administration with their business partners.

A federated model provides the platform for companies to deliver identity-driven transactions to deal with solution extends the user life cycle management of organizations to include trusted business partners and members. Built on open federated SSO and Web services standards, this integrated approach to user life cycle management provides an optimized and cost-effective approach to managing identities and access control rights while simplifying the user experience.

By choosing to operate in business federations, companies:

- ▶ Reduce identity and security management costs through linkage and reuse between companies. Companies no longer need to separately manage users or identities that are not under their control, reducing identity life cycle management costs.
- ▶ Achieve order of magnitude increases in efficiency through reuse of security infrastructure and end-to-end business process integration.
- ▶ Deliver simplified and trusted user experience with single registration, single sign-on because users can navigate easily between Web sites with a single identity and explicitly control release of their personal data. Implement business strategies that drive organic market and customer growth by eliminating the friction caused by incompatible identity and security management between companies.

IBM's federated identity management solution delivers concurrent support for key identity management specifications such as *Liberty*, *WS-Federation* and *SAML*. IBM's federated identity is built on the trust foundation of the WS-Security family of specifications. Integration of federated identity management capabilities

with IBM middleware solutions such as WebSphere enables application platforms to be integrated using industry standards.

In this chapter we have given a view of the business context of federated identity management. We discussed the customer pain points of managing identities. We also described some possible business models where identity federation will bring a real benefit to particular businesses.

The following chapters dive into more details of implementing a FIM solution starting from the architecture and design, followed by how the Tivoli Federated Identity Manager solution and other IBM offerings are responding to this challenge. Part 2, “Customer environment” on page 181, of this book focuses on a few scenarios and how they were implemented.



Architecting an identity federation

A federation is a group of two or more trusted business partners with business and legal agreements, including liability restrictions placed on the 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, be it directly using a Web browser or accessing a local application integrated to an other business partners application. This allows end users to easily accomplish the tasks they need to complete cross-company business transactions. This in turn promotes cross-company business in a loosely coupled environment.

This chapter discusses architecting a federated identity management solution between trusted business partners. It also gives some aspects of understanding how the user life cycle management of identities and the provisioning of user information need to be designed in the federation context.

Briefly, the different standards involved with federated identity management will be described. The end of the chapter briefly explains the on demand Security Reference Architecture and the WebSphere Integration Reference Architecture, and how federated identity management relates to it.

This chapter finishes by taking a look at methodologies related to federated identity management solutions.

The base for discussing the architecting of a solution will be an example used through out this book. Based the example the federated identity architecture is studied, where terminology is explained and finally the specifics of federated identity solutions are addressed.

2.1 Federation example

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

- ▶ An employer, BigCorp, and an employee, Employee One
- ▶ A travel provider, RBTravel
- ▶ A service provider, RBTelco
- ▶ A bank, RBBanking
- ▶ A stock information provider, RBStocks
- ▶ A user John Public coming over the Internet

The involved businesses interact with each other creating a value net of services available to end users, be they public users or employees of a business; see Figure 2-1.

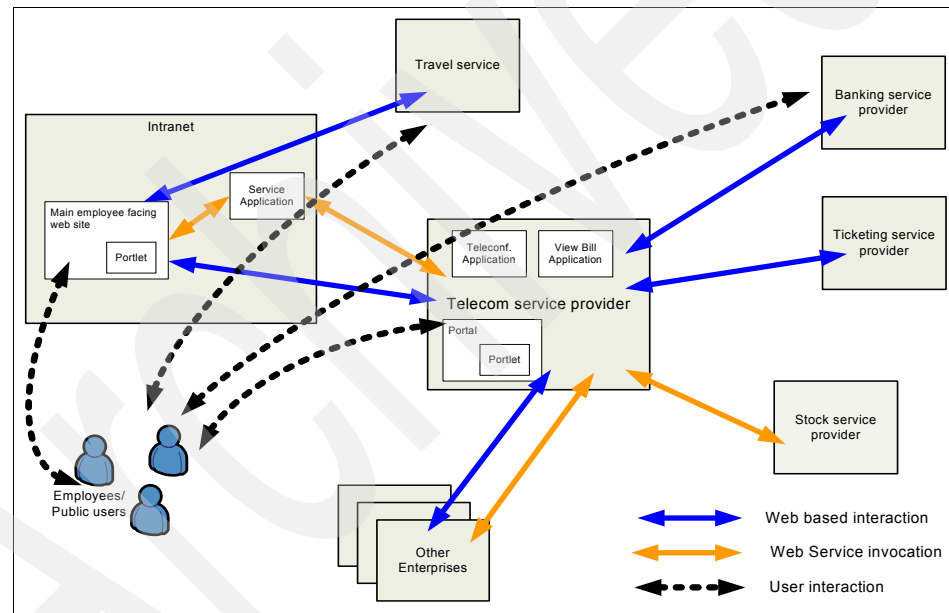


Figure 2-1 Federation example environment

BigCorp

BigCorp is a large company with many employees. As part of providing benefits for its employees, BigCorp 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, BigCorp has outsourced these employee benefits to third-party benefit providers. As

BigCorp is responsible for the management of its users, from account creation (initial hiring) to account deletion/inactivation (dismissal/retirement/other severance), it is natural for BigCorp to continue to assume this functionality but to leverage this in its relationships with third-party benefit providers.

Employee One

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

RBTravel

RBTravel manages travel related services for other businesses, allowing them to order and pay for flights, trains, car rental, hotels and much more. RBTravel have agreements with the businesses using their service to allow anybody from their business who are directed to their Web site to allow them to automatically get an account.

RBTelco

RBTelco is a telecommunications service provider that offers telephony services and also has a portal where RBTelco users or business partner users can choose among offered services to which RBTelco will act as identity provider, offering SSO to the services. RBTelco also has services in their portal connecting to external service partners Web services and presenting them in the portal. RBTelco also acts as a service provider to large enterprises, like BigCorp.

RBBanking

RBBanking offers banking services to its own customers directly and also to RBTelco customers through their portal.

RBStocks

RBStocks offers a stock quote service. The service offers different service levels depending on the user of the service. The stock service is a Web service. RBTelco offers this stock service on their portal.

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

BigCorp enters into a business federation with a travel services provider, RBTravel. RBTravel is to manage a set of services for all of BigCorp's employees. RBTravel is required to manage information about all of these employees as this information is relevant to RBTravel's day-to-day management of the employee travel specific information, like preferences, frequent flier information and so on.

Employee One has an account at BigCorp that he uses to access the BigCorp resources he needs to complete his job. This account is based on his employment at BigCorp. Should Mr. One go on a leave of absence, this account may be suspended. Should he seek employment elsewhere, this account may be terminated.

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

Without federation, Mr. One has to explicitly authenticate to the RBTravel site to access his account even though he has already authenticated to BigCorp and has accessed the RBTravel.com services through his employee portal.

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

From Mr. One's point of view, having RBTravel and BigCorp participate in a federation relationship with reduced sign-on allows Mr. One to authenticate once to BigCorp 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 (BigCorp) and a relying domain (the federated service provider RBTravel) 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 RBTravel is able to participate in a runtime exchange of information with BigCorp which results in some assertion from BigCorp (note that this exchange of information requires no interaction with Mr. One). This assertion is then trusted by RBTravel and used to uniquely identify Mr. One based on an

BigCorp asserted unique identifier. Using this information, RBTravel is able to locally identify and provide access to Mr. One's benefits account information.

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

The second major player in this example is RBTelco. RBTelco 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 may be a service only available to businesses. Attributes that are forwarded from businesses would allow RBTelco to personalize the user experience further if necessary.

Public users to the RBTelco portal would have a personal account. Public users who are customers of RBTelco would benefit from its partners service offerings presented by the portal. The services would allow for reduced sign-on allow the user only to log on to the RBTelco portal and then selecting 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 RBBanking, offering its customers access to their bank services through the RBTelco portal with reduced sign-on, in the same way as RBTravel offered its services to BigCorp employees.

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

Information about the end user, that is necessary for the stock application to be able to deliver the quality of service based information relevant to the users credentials at RBTelco, are included in the request from RBTelco.

2.2 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, identity federation splits into a few different solution areas shown in Figure 2-2. The solution areas are:

- ▶ 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
- ▶ Identity life cycle - Federated provisioning

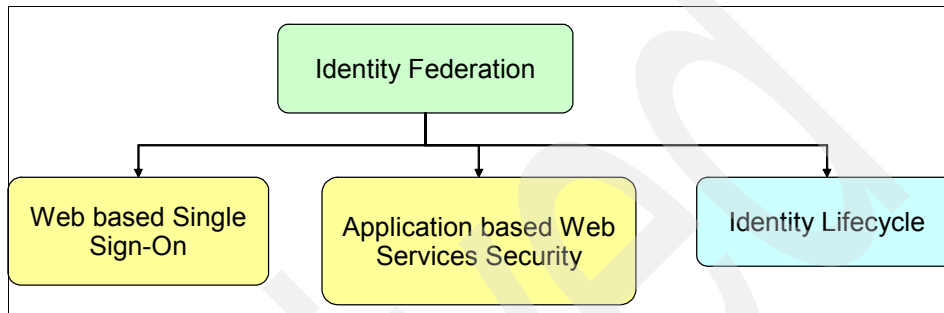


Figure 2-2 Federated Identity Management solution areas

In this section the common fundamentals and terminology for the three solution areas will be covered, starting with a background on FIM, an architecture overview and finishing of with the general architectural FIM terminology and concepts.

In the sections following this one, the specifics of the three solution areas will be studied in a little more detail with regards to their functional specifics, 2.4, “Federated single sign-on” on page 59; 2.5, “Web services security management” on page 65; and 2.6, “Federated identity provisioning” on page 70.

Chapter 3, “Tivoli Federated Identity Manager architecture” on page 85, will do the same, but look at the IBM Tivoli Federated Identity product specific approach.

2.2.1 Background to federation

Federation solutions are successful when they allow customers, business partners, and end 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 growing administrative cost due to that 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, as we discussed in 1.2, “Business environment” on page 5; and 1.3, “IT environment” on page 17.

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
- ▶ Off load 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 business partners need 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 find it extremely difficult to engage in a federated model when it does not have the same visibility of life cycle management of third-party users as they do with their 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 questions 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, such that the federated relationships look like an extension of their existing identity management procedures.

2.2.2 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 functionality:

- ▶ Seamless and secure user interaction across federation business partners (aka, *federated single sign-on*)
- ▶ Seamless and secure business interaction across application platform integration (aka, *Web services security management* for Service Oriented Architectures)

Both of these functionality 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 2-3. Both federated single sign-on and Web services security solutions are built on a trust infrastructure.

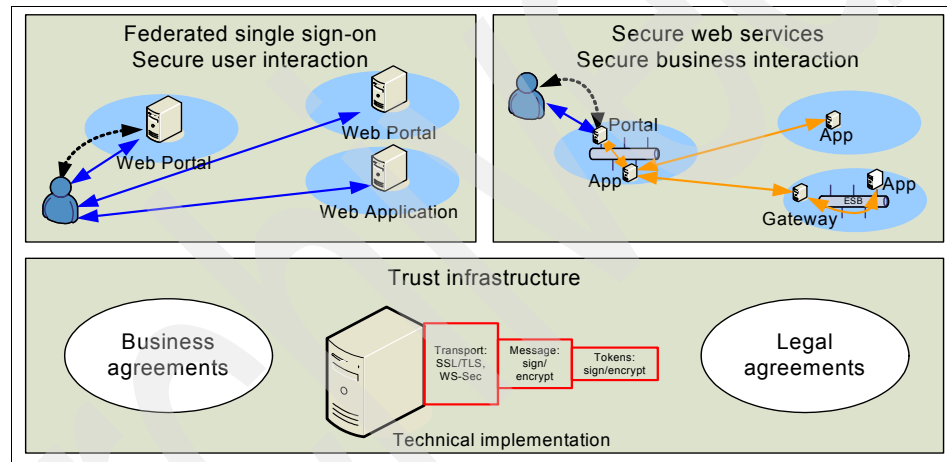


Figure 2-3 Base trust infrastructure for secure services

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 just single sign-on functionality. Standards and specifications such as the SAML specification and WS-Federation and Liberty Alliance ID-FF 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 their 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 inter-operability 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 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 (be it user or application) is a form of reduced-sign-on.

Unlike the federated identity management single sign-on described above, however, this 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 yet complementary) solutions: *Federated single sign-on* and *Web services security management*.

To design a solution, the following areas need to be understood, and are covered in this section:

- ▶ The *roles* of identity and service provider: The definition who is the authoritative source of the user identity information
- ▶ Identity/attribute *mapping*: The definition of the attributes to be shared and the mapping of them in the business partner systems
- ▶ User account *management/provisioning*: The procedures for managing user identity data, agree what information can be shared, and what information is independently managed by users, and will the users be provisioned automatically to the new endpoint (a priori or 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/de-provisioning procedures
- ▶ *Trust*: The process of ensuring security for connections/transport, messages and tokens

- ▶ Selection of the federation protocol profile(s): The definition of the federation protocol profiles to be used between the two business partners

2.2.3 Roles

Within a federation, business partners play one of two roles: *Identity provider* or *service provider* or both. The identity provider (IdP) is the authoritative site responsible for authenticating an end 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 2-4. 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.

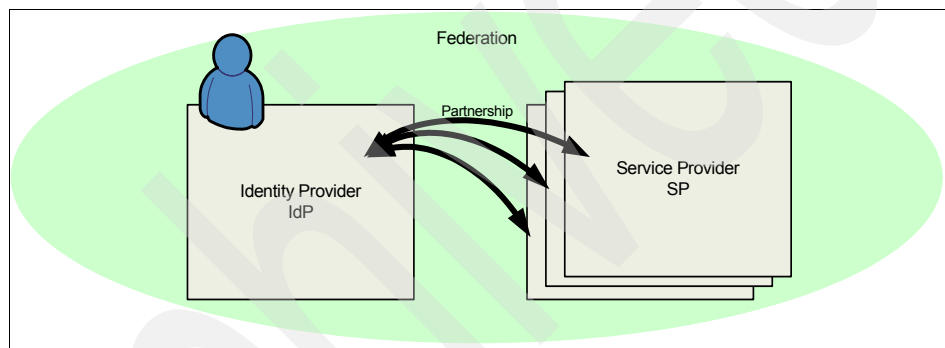


Figure 2-4 Identity provider and service provider in the federated model

Identity provider - IdP

The identity provider is responsible for account creation, provisioning, password management, and general account management, and also acts as a collection point or 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). These service providers will leverage their trust relationships with an IdP to accept and trust vouch-for information provided by an IdP on behalf of a user, without the direct involvement of the user. This enables businesses (service providers) to off load identity and access management costs to business partners within the federation.

In the example in 2.1, "Federation example" on page 33, both BigCorp and RBTelco act as an identity providers. RBTelco is also a service provider.

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 - SP

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

2.2.4 Identity models

Shared and distinct identity models refer to the nature of the identity data management. A shared identity data management solution implies that information can be managed by one business partner (the identity provider). Distinct identity data management solutions imply that information is replicated across business partners and managed separately across business partners.

Shared

A shared identity approach to federated business interactions may be appropriate when one business partner is able to trust and rely on the assertion of a user's identity data by an identity provider. In this model, federation allows the user (and the federation business partners) to establish a common unique identifier, used to refer to the user. Based on this common identifier, an identity provider is able to manage a user's identity data, acting as the authoritative source of this information to trusted service providers.

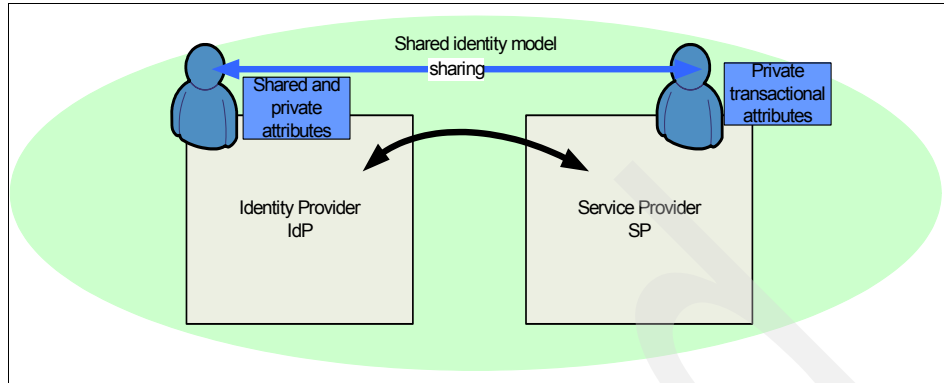


Figure 2-5 Shared identity model

The fundamental question with respect to identity and attribute provisioning between business partners is what information can they share and what are the benefits of sharing? In an optimistic scenario an IdP and SP share every piece of information about the user as in Figure 2-5.

- ▶ *Sharing authentication credentials* between the identity provider and service provider means that the service provider can rely upon the identity provider to authenticate the user. This frees up the service provider from managing the password and credentials for the user. If identity account data cannot be shared then both identity provider and service provider must manage a separate identity account for the user, forcing the user to remember multiple accounts and passwords.

Note: Regardless of the sharing of account data, both an identity provider and service provider will usually maintain (at least) a set of transactional attributes associated with a user's identity.

- ▶ *Sharing transactional attributes* requires that the identity provider and service provider agree upon the roles and entitlements or groups that the user belongs to up front. This is a difficult proposition to implement, as two independent providers typically have different ways to group identities or manage role information. Rather than sharing transactional attributes, a provider may map their transactional attributes in a form that their business partner understands. In this approach identity meta-data is maintained separately at both identity provider and service provider.
- ▶ *Sharing profile attributes* between identity provider and service provider is usually a function of user consent. This is more dictated by user preference and user privacy concerns. Sharing of attributes in many cases will require user consent (OPT in) and the ability to prove user consent. In a pragmatic

sense, some attributes may be shared (such as e-mail address), whereas some attributes will not be shared. If attributes cannot be shared then the attributes need to be replicated between the identity provider and service provider. So if, for example, a user's home address is replicated, both business partners must independently manage this information. If the user moves, and one of the business partners knows about the updated address, in a distinct identity model, the business partner cannot notify/provision this information to other business partners.

Provisioning plays a key role in determining all three of the above scenarios when the identity information cannot be shared between IdP and SP. This will be discussed in more detail in 2.2.6, "Trust" on page 49 and 2.6, "Federated identity provisioning" on page 70.

Distinct

A distinct identity approach to federated business interactions may be appropriate when the two organizations cannot share identity information. This may happen because of anti-competitive practices, separation of data, dis-intermediation (companies unwilling to share customer data with business partners for competitive reasons), political reasons, or because the user has an independent relationship with both providers.

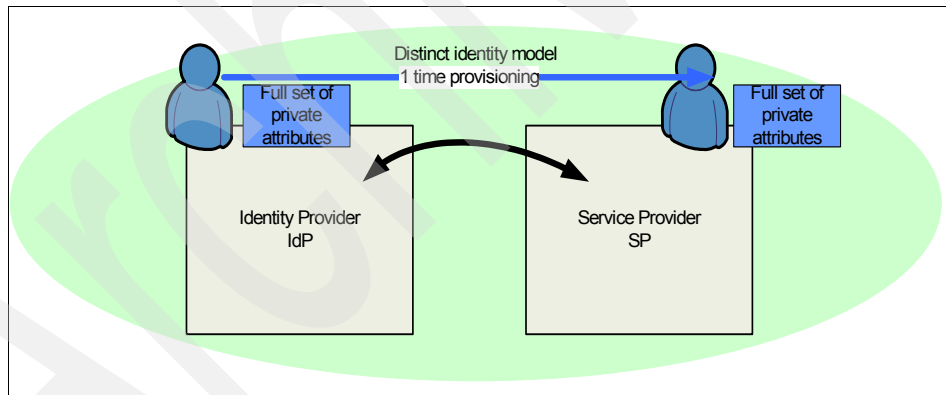


Figure 2-6 Distinct identity model

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 as in Figure 2-6.

2.2.5 Identity attributes

In a federated model an identity provider and service provider need 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:

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

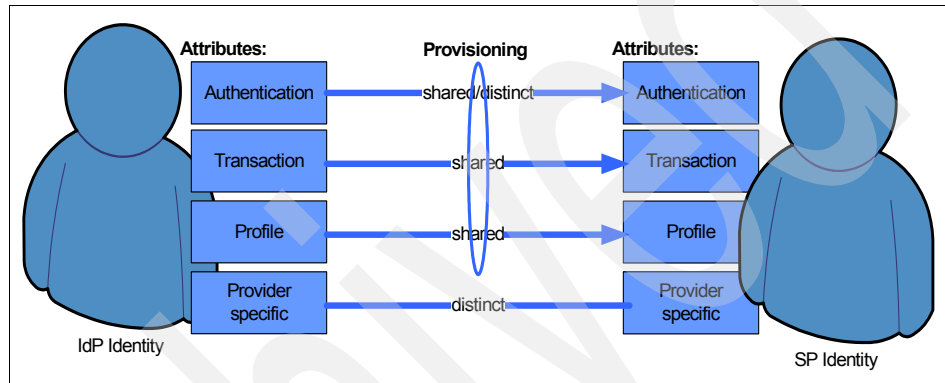


Figure 2-7 Shared and distinct identity data and attributes

For each class of *identity data*, we can allow for a *shared* or *distinct* identity data management solution as shown in Figure 2-7. 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 some other 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. Thus 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 may 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 there is no need to provision authentication credentials. 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.

There may be some cases where this is not 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 include 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 may come in one of two flavors: Runtime (or just-in-time) and a priori provisioning, described in 2.6, "Federated identity provisioning" on page 70.

Transactional attributes

Transactional attributes include information that describes a user and his affiliations and entitlements. This information is bound to a user's identifier. This may include groups that the user belongs to or roles that he can assume. This data may 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 authorization/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/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 whenever 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/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 end 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 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 state of residence. 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 BigCorp employee, Mr. One, who participates in a frequent flier program with his airline of choice. Mr. One has an online travel account at RBTravel that he uses to book his air travel; this account is bound to his identity. Associated with this user name is Mr. One's password (authentication credentials) used to authenticate, these are not known by Mr. One because they were setup as part of his provisioning from BigCorp. Associated with Mr. One's travel account, are Mr. One's profile attributes (for example, his billing address, e-mail, telephone number).

Based on Mr. One's travel account, the travel service will assign (and manage) Mr. One's frequent flier status (a transactional attribute). When attempting to book a flight Mr. One's attributes will be used to assist him in booking the flight and also enable the ticket to be issued to his frequent flier card. When Mr. One attempts to book a trip, his travel class may be based on attributes with regards to his airline points or position at BigCorp. Once Mr. One has selected his desired travel and is about to book it, secondary evaluating of Mr. One's identity will be accomplished as part of the specification of Mr. 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 end-user* by managing their profile information at their 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 may 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 may include a user's preference to always search for new auction items within the "Toys less than \$25" category.

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.

2.2.6 Trust

Trust is a key capability for all three solution areas, and therefore a key area for FIM. Trust services are also discussed in some detail in 3.2.3, “Trust services” on page 92.

A trust relationship is represented at a technical level by cryptographic keys 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 need to communicate information about a user in a secure and trusted fashion. This is accomplished by leveraging a trust infrastructure.

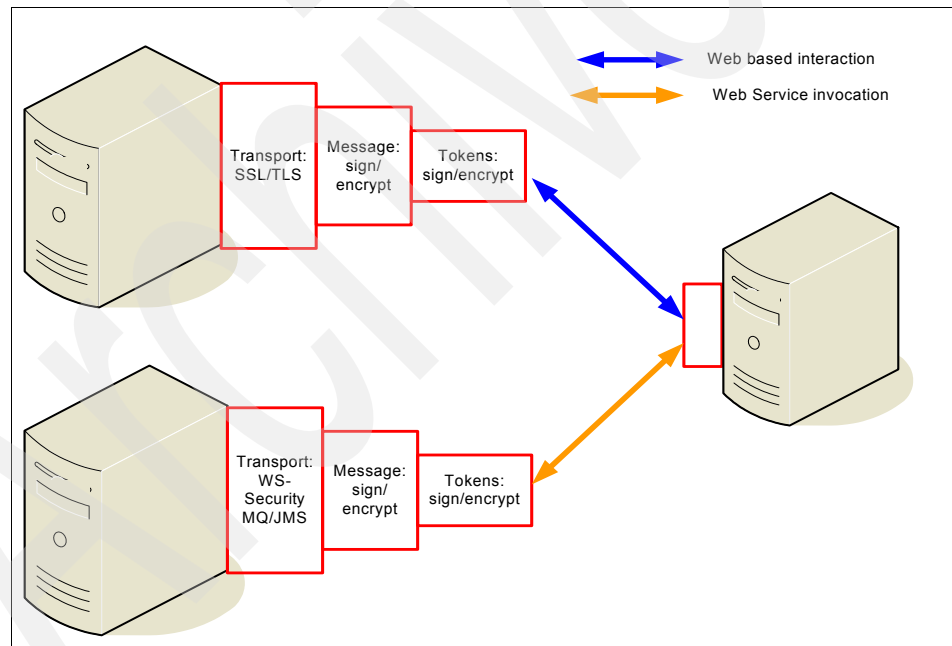


Figure 2-8 Layers of trust

A trust infrastructure enables the protection of a message at all levels, Figure 2-8 on page 49:

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 a FIM flow

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

Transport

The simplest form of trust infrastructure is that provided by the transport layer SSL protocol, 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, may not always run over SSL-compatible transport protocols; Web services may be invoked via 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 may be applied.

Federated identity management requests will 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 may 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 and/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/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.

2.2.7 Federation protocol

When creating a federation an agreement needs 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 may do the same, but one needs to be defined for each federation partnership.

The different standards and efforts in this space are discussed in 2.3, “FIM standards and efforts” on page 51. The different standards have different capabilities that will govern the choice of protocol, made. There is a table in 2.3.9, “Selecting Federation standards” on page 58, that may be used to help select SSO protocol.

2.3 FIM standards and efforts

Reduced sign-on techniques and solutions have been in place for many years now. Federated identity management has its roots in reduced sign-on technologies. IBM Tivoli first introduced reduced sign-on support in Tivoli Access Manager as early as 2001.

The first standards-based efforts in this space were by Internet (*Shibboleth*) and OASIS (*SAML*). Since then, federation efforts have been led by the Liberty

Alliance (*Liberty ID-FF*) and through the Web services work of IBM and Microsoft and partners (*WS-Federation*). Each of these efforts is introduced and briefly discussed in the following sections. The more recent Web services standards including WS-Security, WS-Trust and WS-Provisioning are presented as well.

2.3.1 SSL/TSL

Secure Sockets Layer (SSL, standardized as Transport Layer Security, TSL) provides session-level security through the use of encryption. While 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 some 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 need to incur the overhead of encryption/decryption. |
| End-to-end | SSL protection ends if intermediate components need to examine transactions. No provision is made for encrypting end-to-end across intervening components. |

Web services security (discussed elsewhere in this section), however, overcomes these issues.

2.3.2 Security Assertion Markup Language (SAML)

Security Assertions Markup Language is a specification designed to provide cross-vendor single-sign-on interoperability. SAML was developed by a consortium of vendors (including IBM) under the auspices of OASIS, through the OASIS SSTC (Security Services Technical Council). SAML has *two major components*: It describes *SAML assertions* used to transfer information within a single sign-on protocol and *SAML bindings and profiles* for a single sign-on protocol.

A SAML assertion is an XML-formatted token that is used to transfer user identity (and attribute) information from a user's identity provider to trusted service providers as part of the completion of a single sign-on request. A SAML assertion provides a vendor-neutral means of transferring information between federation

business partners. As such, SAML assertions have a lot of traction in the overall federation space.

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 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 functionality, 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 takes as input 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. SAML 2.0 is still largely in development with customer adoption/deployment expected to take off in mid-2006.

More information on the SAML specification is available from:

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

There is also more information about SAML in 3.3.5, “F-SSO approaches” on page 110.

2.3.3 Shibboleth

Shibboleth is related to SAML but is specific to the higher-education sector. Shibboleth uses some of the SAML protocols but includes additional features specific to the higher-education community. Shibboleth introduces the notion of Where are You From? processing, allowing a service provider to implement both push-based and pull-based SSO protocols. Shibboleth has been submitted as a contributor to the SAML 2.0 specification.

For example, within the higher-education community, there are very strict rules on the release of information about an institution's students, even to other higher-education institutions.

2.3.4 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. For more information about Liberty Alliance, see:

<http://www.projectliberty.org>

The *Liberty Identify Framework, ID-FF*, describes federation functionality that goes beyond single sign-on. Initially released as Liberty Alliance ID-FF 1.0 in July 2002, the latest release of the Liberty specification is Version 1.2, released November 2003.

Liberty ID-FF describes profiles for B2C-based single sign-on and additional functionality. Liberty ID-FF profiles include: *Single sign-on (SSO)*, *single log-out (SLO)*, *Account Linking (Register Name Identifier, or RNI in ID-FF 1.1)*, *Account De-Linking (Federation Termination Notification, or FTN in ID-FF 1.1)*, and *identity provider introduction (IPI)*. The Liberty-specified common user identifier (CUID) is referred to as a *NameIdentifier*. It is an opaque reference to a user that acts as an *alias*, meaning that it cannot be used to infer information about the user, such as her identity. A Liberty *NameIdentifier* is used to establish (and maintain) the account linking between an IdP and an SP. The RNI profile is used to allow a reset of a user's *NameIdentifier*, replacing a current value with a new *NameIdentifier* value. The FTN process is used to remove all references to a *NameIdentifier*, thus achieving account de-linking. Taken together, these profiles are intended to provide richer user management functionality within a federation than simple single-sign-on.

In ID-FF 1.2, the RNI and FTN profiles have been collapsed into a single profile, the *Manage Name Identifier (MNI)* profile. This profile moves all of the account linking life cycle into a single profile.

The Liberty approach is based on business affiliates forming *circles of trust*. The Liberty circles of trust is defined as “a group of service providers that share linked identities and have pertinent business agreements in place regarding how to do business and interact with identities.”

This is an excerpt from:

<http://www.projectliberty.org/about/faq.php#07>

There is also more information about Liberty ID-FF in 3.3.5, “F-SSO approaches” on page 110.

2.3.5 WS-Federation

WS-Federation is a specification defined by IBM, Microsoft, VeriSign, and RSA within the scope of the IBM-Microsoft Web services security roadmap.

WS-Federation was published on July 8, 2003. WS-Federation interoperability between IBM and Microsoft has been demonstrated several times, including by Bill Gates and Steve Mills in New York City in September of 2003. Subsequent to that, a public interoperability exercise was held on March 29–30, 2004 between IBM, Microsoft, and other third-party vendors.

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

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.

WS-Federation Active 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.

WS-Federation Passive 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 *vanilla HTTP browser*. WS-Fed 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 three specifications that make up WS-Federation are available for download from IBM DeveloperWorks at:

► WS-FED:

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

▶ WS-FEDACT:

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

▶ WS-FEDPASS:

<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 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 vanilla browsers, enhanced browsers, active clients, and application-to-application connectivity.

There is also more information about WS-Federation in 3.3.5, “F-SSO approaches” on page 110.

2.3.6 WS-Trust

The Web Services Trust Language (WS-Trust) uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the issuance and dissemination of credentials within different trust domains.

In order to secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can *trust* the asserted credentials of the other party. This specification defines extensions to WS-Security for issuing and exchanging security tokens and ways to establish and access the presence of trust relationships. Using these extensions, applications can engage in secure communication designed to work with the general Web Services framework, including WSDL service descriptions, UDDI businessServices and bindingTemplates, and SOAP messages.

The specification that makes up WS-Trust is available for download from IBM DeveloperWorks at:

<http://www.ibm.com/developerworks/webservices/library/specification/ws-trust/>

2.3.7 WS-Security

While 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 the next section).

The OASIS Security Services Technical Council, together with the OASIS Web Services Security Technical Council, 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 on the OASIS Web services security specification is available from:

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

There is also more information about WS-Security in 3.4.2, “WS-Security” on page 125.

2.3.8 WS-Provisioning

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.

The WS-Provisioning interface is an open standard that is available to other companies that want to develop interoperable provisioning scenarios and systems. The specification is publicly available on the IBM developerWorks Web site:

<http://www.ibm.com/developerworks/webservices/library/ws-provis/>

WS-Provisioning has been submitted to the Organization for the Advancement of Structured Information Standards (OASIS) Provisioning Service Technical Committee.

Tivoli Federated Identity Manager supports draft Version 0.7 of the WS-Provisioning specification.

This is an excerpt from the IBM DeveloperWorks definition of WS-Provisioning.

There is also more information about WS-Provisioning in 3.5, “Provisioning services” on page 129.

2.3.9 Selecting Federation standards

To help in selecting which F-SSO profile to use, see Table 2-1.

Table 2-1 highlights some of the characteristics of each protocol: SAML 1.0 and 1.1 (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 2-1 Characteristics per 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
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 log out (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

Supported characteristic	SAML 1.0, 1.1	SAML 2.0	Liberty ID-FF 1.0, 1.1, 1.2	WS-Federation
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. While 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 may not require this (see item 9 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.

You can find more information about SAML, Liberty ID-FF, and WS-Federation in 3.3, “Federated single sign-on” on page 100.

2.4 Federated single sign-on

Federated single sign-on 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/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 (see 2.3.9, “Selecting Federation standards” on page 58). 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.

In this section Federated single sign-on functionality is discussed, this is also studied more in detail, out of an IBM Tivoli Federated Identity Management product point of view, in 3.3, “Federated single sign-on” on page 100.

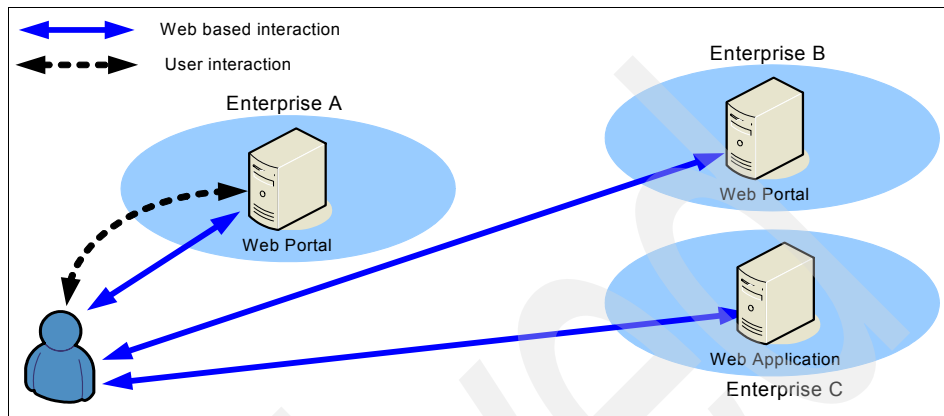


Figure 2-9 Secure user interaction - F-SSO

A simplified view of a user interaction is illustrated in Figure 2-9. where a user interacts with Enterprise A who acts as the IdP and two businesses Enterprise B and C who act as SP's. The user interactions are all Web browser based and F-SSO is used to reduce sign-on for the user. The reduced sign-on may be accomplished with any of the SSO protocols, SAML, Liberty ID-FF, or WS-Federation; see Table 2-1 on page 58 for help with selecting SSO protocol suitable for the federation partnership to be set up.

In the attempt to explain the different functionality in Federated single sign-on, the example in 2.1, “Federation example” on page 33 will be used in this section.

Functionality relevant to F-SSO are; pull and push SSO protocols, account linking, WAYF, session management, logout, credential clean up, global good-bye and account de-linking.

2.4.1 Push and Pull SSO

There are two different ways of doing SSO, push and pull. Pull SSO is available in SAML 1.x and 2.0, Liberty ID-FF and WS-Federation. Push is available in SAML 1.x (with custom coding in Liberty ID-FF) and WS-Federation; see 2.3, “FIM standards and efforts” on page 51, for details.

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.

BigCorp uses pull SSO when its employees sign on to RBTravel.

2.4.2 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. Once accounts are linked, the user can authenticate to one provider and then navigate seamlessly to various service providers with whom they have linked accounts without having to re-authenticate or enroll.

At a technical level account linking is the process by which an identity provider and service provider agree on some common unique identifier, and then each bind their internal, local user identity to this *common unique identifier* (CUID). This allows the identity provider and service provider to refer to the user by their CUID during single sign-on without disclosing information about their local internal representation of the user.

Consider RBTelco and RBBanking, where John Public has distinct (authenticate-able) identity accounts at each company. When the two companies agree to join a federation, they must somehow enable RB Telco's users for SSO to RBBanking. In general, this will happen based on functionality at RBBanking. This happens through a two-step process, in this case initiated from the RBTelco site. RBTelco changes the functionality at the portal, so that instead of a simple redirection to RBBanking, the clicking of a link to RBBanking initiates single sign-on to RBBanking. RBBanking receives this single sign-on request but is not able to map the user to a locally known user. This will cause RBBanking to prompt John for his RBBanking authentication credentials. Successful authentication by John will now give RBBanking the RBTelco asserted CUID (from the SSO request) and its own local representation of the user (from John's direct authentication). RBBanking is now able to establish the account linking that will allow this user to SSO from RBTelco.

Should users directly access RBBanking during the roll-over period, they will be authenticated by RBBanking as usual. After this, RBBanking will request SSO from RBTelco (for the already authenticated user). The corresponding SSO

response will contain the common user identifier (CUID) so that RBBanking has the RBTelco asserted CUID (from the SSO request) and its own local representation of the user (from John's direct authentication). RBBanking.com is now able to establish the account linking that will allow this user to SSO from RBTelco.

RBBanking may choose to disable the user's local password, so that direct authentication to RBBanking is no longer possible as long as the user's account is linked with RBTelco. The next time this user attempts to directly access RBBanking, RBBanking will request an SSO from RBTelco.

Part of the account linking process is normally the establishment of some long-term/persistent piece of information, such as an HTTP cookie, that identifies RBTelco as this user's identity provider. During the roll-over period, this is also used to distinguish between already linked and yet-to-be-linked users from RBTelco. Once the roll-over period has completed, all users without this persistent information must be queried to determine if RBTelco really is their identity provider (see the following section for more information). In use case 3 this scenario is shown in some detail, see Chapter 9, "Use case 3 - Liberty" on page 245.

2.4.3 Where are you from (WAYF)

Some service providers may have trust relationships with multiple identity providers. This means that a user may possibly initiate SSO from one of many IdP's. 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.

In the case of RBBanking, the WAYF information is established during the roll-over period. During the roll-over period, RBBanking is acting as both a service provider (for already federated users) and an identity provider (for not yet federated users). That is, both RBBanking and RBTelco are acting as identity providers for the single service provider, RBBanking.

If RBBanking was a SP to several IdP's, it must rely on some form of persistent information associated with a user (such as an HTTP cookie) to identify to which identity provider an SSO request is to be directed. If this cookie is absent, then RBBanking must engage in some form of user-interactive WAYF processing. RBBanking may choose to prompt John to select such an identity provider from a list of known/trusted identity providers.

In some cases, a service provider may not be willing to expose a list of trusted identity providers. In this case, John would be given instructions by RBBanking to directly access his identity provider (RBTelco) and initiate a SSO request through an identity provider based mechanism.

While this does involve a level of interaction with the user, neither situation is as intrusive as requiring that the user remember a password for RBBanking. Ideally, user-interactive WAYF processing should not be required every time John accesses RBBanking.

2.4.4 Session management and access rights

Once a user has single signed-on to a service provider, the SP is responsible for managing the user's local session at the SP. This includes authorization decisions on the user's requested actions and also session management itself, such as logoff or session time-out.

This implies 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 may 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.

2.4.5 Logout

In some federation scenarios, the notion of global logout (single sign-off) 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 SSO:ed in a given session. The IdP will then send a logout request to each of these SPs on behalf of the user.

It may be the case, for example, that if John logs off of RBTelco's portal, that RBTelco is no longer willing to honor any transactions that John may undertake as a result of his RBTelco vouched SSO actions. In this case, RBTelco will trigger a logout request to all business partners to which an SSO request has been issued within John's current session.

Global logout does not imply that local logout goes away. It is possible that a user will wish to log out of a session at a service provider without destroying their session at the identity provider. Note that this 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/inactivity time out than is used in a standard, directly authenticated session. A shorter inactivity time out for an SSO user may be acceptable, as the user is not forced to explicitly re-authenticate. Instead, the SP will simply re-request an SSO from the user's IdP.

2.4.6 Credentials clean up

Logout, be it 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 may be independent of sessions with back-end applications. Back-end application sessions may be used to maintain a state between request/responses of a multi-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.

Consider what happens when John logs out of the RBTelco portal and is single logged-out of the RBBanking site. If John had started a transaction (to transfer assets, for example) and then forgotten about this, this transaction needs to be cleaned up (this is a form of garbage collection). If this does not happen, RBBanking may be left with orphaned sessions that can tie up resources at its back-end applications.

2.4.7 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 may be terminated in several ways: A user may choose to terminate his binding of an identity provider to a service provider or an IdP and SP may choose to no longer do business together, breaking the binding for all of the IdP's users.

For example, consider Employee One as an employee of BigCorp. If Mr. One changes employers (now working for SmallCo), Mr. One's access rights and entitlements to BigCorp's sponsored travel rates must be cleaned up as part of the global good-bye between BigCorp and RBTravel. Note that global good-bye does not imply that Mr. One's account, including provider-specific attributes, at RBTravel is removed. It simply implies that all of the BigCorp attributes, including BigCorp-relevant transactional and profile attributes, are de-provisioned (deleted) from Mr. One's account at RBTravel.

In general, global good-bye is accomplished together with account de-linking.

2.4.8 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/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 service provider may choose to link an account with a different identity provider, or the SP may choose to resume direct authentication of the user.

At some point, John Public may choose to close his RBTelco account. This may happen because John moves or changes network provider, and so on. In this case, John is no longer able to SSO to RBBanking from RBTelco because he is no longer able to sign on to RBTelco. In this case, John's information at RBTelco and RBBanking should be de-linked (sometimes referred to as de-federated). The result of this process will be that the common, unique identifier for John will be destroyed, the ability of John to single sign-on from RBTelco will be lost, and John will be reinstated as a user who is able to directly authenticate to RBBanking (in turn implying some form of self-registration process by RBBanking to allow John to re/set a password for RBBanking).

In use case 3 this scenario is shown in some detail, see Chapter 9, "Use case 3 - Liberty" on page 245.

2.5 Web services security management

Businesses need a standard way for service requestors (suppliers, customers and partners) to securely find the right Web services of a given business. Business service providers need to be able to securely identify and expose the right Web service to only authorized requestors.

Web services security management functionality allows the establishment and management of federation relationships for application to application interactions, see Figure 2-10 on page 66, enabling the required trust and security. In this solution an application, is able to generate a Web services request, acting as a Web services client. This request can then be secured (encrypted and signed) to provide message-level confidentiality and integrity.

Web services security management provides the key capability to be able to realize a service oriented architecture, where businesses seamlessly and dynamically interact with each other as part of new horizontally integrated process.

Web services security management adds the ability for message-level authentication and authorization, in the context of a federation relationship. This is studied in detail, out of a IBM Tivoli Federated Identity Management product point of view, in Example 3.4 on page 121.

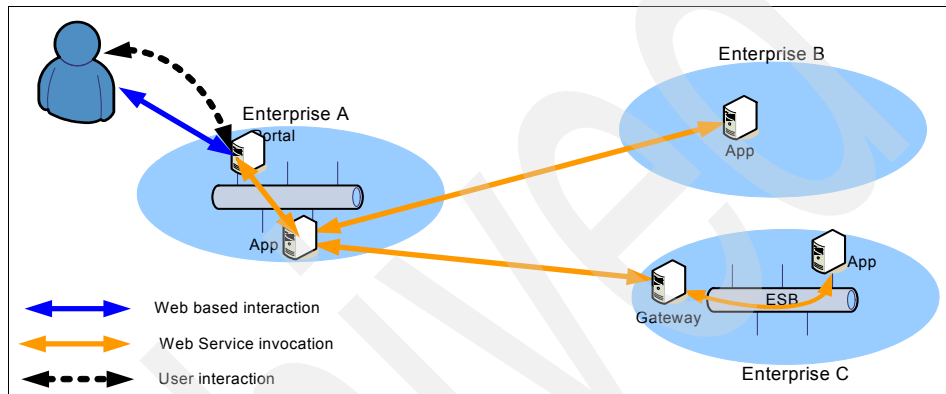


Figure 2-10 Secure business interaction - Federated Web services security

A simplified view of a user interaction is illustrated in Figure 2-10. where a user interacts with the portal in Enterprise A. The portal renders an application which uses Web services to integrate with any of the two businesses Enterprise B and C who have exposed an application as a Web service. The interactions are all application to application based and Web services security management is used to enable security in the end to end integration.

While explaining the different functionality in Web services security management, the example in 2.1, “Federation example” on page 33, will be used in this section.

Functionality relevant to Web services security management are Web services, WS-Security, and gateways.

2.5.1 Web services

Web services have emerged as the most promising development to address cross-enterprise, platform, and vendor business integration issues. Web services is a family of emerging technologies that enable easy interoperability of

programmed IT services and integration of applications into a businesses increasingly horizontal business processes.

Web services technology enables businesses to describe available services and provide access to those services over standard Web protocols and communications boundaries. Web services has inherited and learned from the way the World Wide Web revolutionized how people talk to systems. The new customers and business models, extensions of opportunity, new transparency and improved collaboration within enterprises and in some cases simplification in infrastructure and sometimes reduced cost. The key to these successes was a general server-to-client model in a highly distributed environment, and most importantly based on simple open standards and industry support.

Web services promises to do the same thing for the way systems talk to systems: integrating one business directly with another. This should be done in a dynamic way without waiting for human intervention. It is about getting your own business talking to itself or your suppliers, customers or partners, to provide integrated IT systems, with the potential for dramatic reductions in infrastructure complexity and costs. The key, here as well, is a general application-to-application communication model based on simple open standards and industry support.

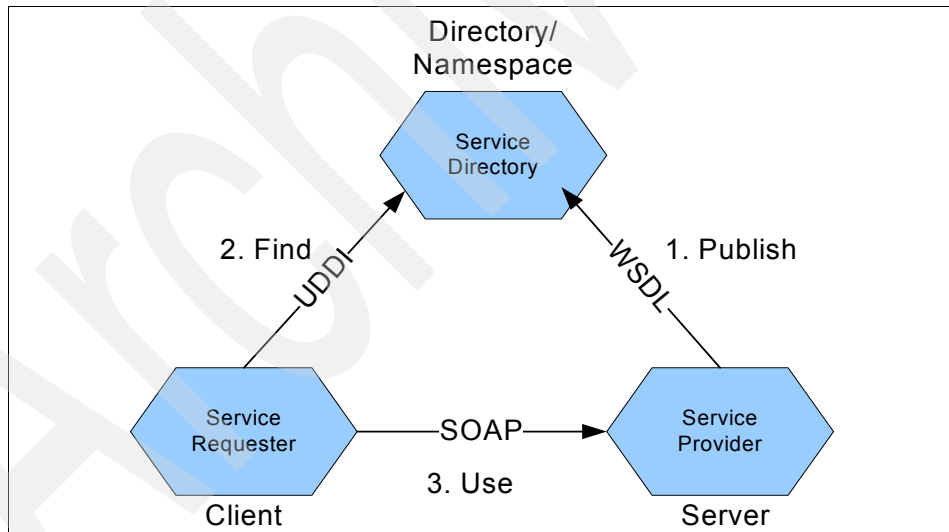


Figure 2-11 Basic Web services

Figure 2-11 shows the basic interaction model supported by Web services. Basic Web services define interactions among service requesters, service providers, and service directories as follows.

Service requesters find Web services in a UDDI service directory. They retrieve WSDL descriptions of Web services offered by service providers, who previously published those descriptions to the service directory. After the WSDL has been retrieved, the service requester binds to the service provider by invoking the service through SOAP.

When a user like John Public access RBTelco to view his stock service, RBTelco uses a Java application to collect the stock information from RBStocks and present it in the portal. The application at RBTelco then acts as a Web service service requester making a SOAP request to the service prover RBStocks who based on the passed identity and attributes returns the requested data.

The basic Web services are often described in terms of SOAP, WSDL, and UDDI. However, it should be noted that each of these standards can be used in isolation, and there are many successful implementations of SOAP alone, or SOAP and WSDL, in particular.

For more information on Web services see the Redbook *Using Web Services for Business Integration*, SG24-6583-00, or Web services architecture - W3C Working Draft:

<http://www.w3.org/TR/ws-arch/>

2.5.2 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. All this with the goal to accomplish end-to-end message content security.

For more on 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 2-12 on page 69, 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

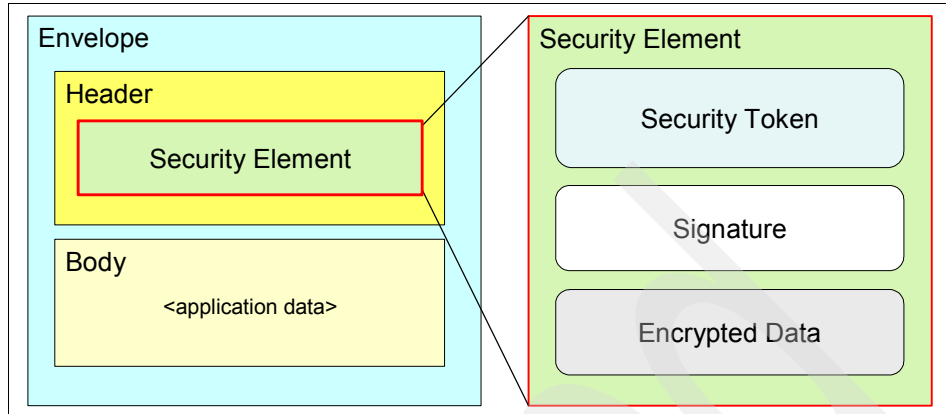


Figure 2-12 WS-Security: SOAP message security, extensions to the header

These elements can be used to achieve *message-based security* for a SOAP message. That is, the message in and of 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.

When RBTelco securely passes the client identity and attribute information to RBStocks, 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 allows for the request to be honored by the federated Web service hosted at RBStocks by having the token processed by RBStocks, including the verification, user ID and attribute mapping, authorization, and token transformation that is associated with being a security token consumer.

2.5.3 Gateways

A Web services gateway or firewall is much the same as a HTTP Reverse proxy. A WS Gateway enables the company to separate internal network topology from the Internet allowing for flexibility and abstraction.

A Web services Gateway addresses the de-coupling of deployment from invocation, process abstraction, flexibility and protocol transformation, much as any network gateway will in its functional area. See 3.4.3, "Web services Gateway or Firewall" on page 126, for more on the functionality of the gateway.

RBTelco has a Web services gateway which it uses when the application server needs to pass client identity and attribute information to an external application at for example RBStocks. The gateway then, on the outbound side, adds a SAML assertion as a security token in a Web services request allowing that request to be honored by the federated Web service hosted at RBStocks.

In 4.2.3, “XML gateway pattern” on page 155, there is more on Gateways and are some examples of Gateways available in the market.

2.6 Federated identity provisioning

Provisioning is about remotely having the capability of managing attributes of for example a user as part of an identity management process. The same provisioning definition is also valid for provisioning of other services or resources for example applications or servers. Within federated identity management the focus is on the user/identity. This is studied in more detail, out of a IBM Tivoli Federated Identity Management product point of view, in 3.5, “Provisioning services” on page 129.

In the attempt to explain the different functionality in provisioning, the example in 2.1, “Federation example” on page 33, will be used in this section.

Federated identity provisioning extends these provisioning management activities beyond an internal trust domain, see Figure 2-13 on page 72. Federated identity provisioning makes it possible to extend 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.

When used with provisioning of account data and authentication credentials, provisioning solutions generally come in one of two flavors: *Runtime* (or just-in-time) and *a priori provisioning*. Runtime provisioning solutions are also referred to as enrollment solutions as 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 allows both the identity provider and service provider to create local accounts/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 identity provider. A priori provisioning may also be triggered by other events, such as a

change in a user's status that in turn gives him access to more business partner resources, or a subscription event by a user, signing up for services that the identity provider in turn outsources to a third-party service provider. Note that like 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/record in her local registry in response to a single-sign-on request from a trusted identity provider. This may happen when an SP receives a SSO request from a trusted identity provider but does not have any record of the user claimed in the SSO request. Instead of rejecting the SSO request, the SP may 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.

In the case of BigCorp, provisioning a new employee within the BigCorp system will cause account creation of the user's BigCorp required accounts. A federated provisioning solution could also cause the sending of a provisioning trigger request to RBTravel, but in this case just-in-time provisioning is used instead and the user is provisioned at RBTravel on the fly if no user exists there. As this account is created during the single sign-on from the user in BigCorp, the common user identifier information will have been included with the provisioning request and so no account linking step is required by this new 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 end user's identity provider. In the case of BigCorp, employee Mr. Employee One may have a corporate credit card used for travel purposes. If this credit card number changes, BigCorp may be required to provision this transactional attribute to BigCorp's travel agency RBTravel. Similarly, Mr. One's salary may be considered a transactional attribute, as it will be used by benefits providers to determine Mr. One's eligibility for services. As such, it must be provisioned to BigCorp's benefits providers if/when it changes.

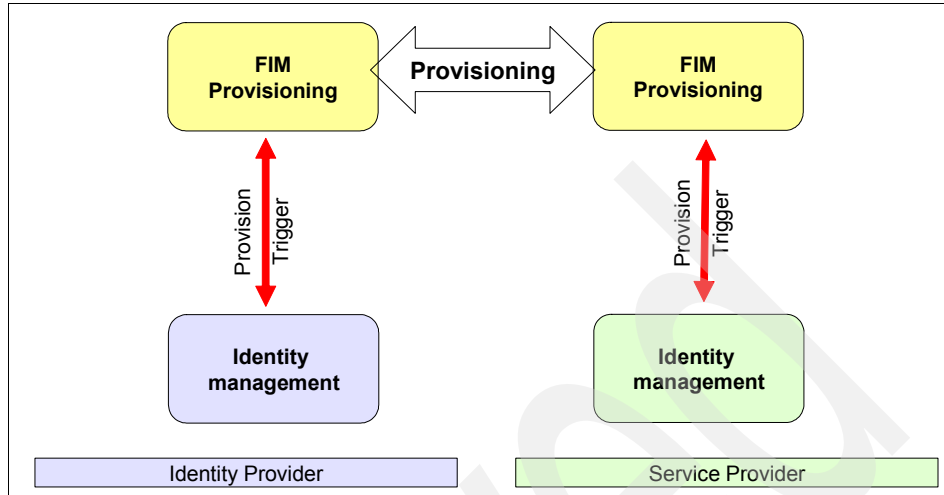


Figure 2-13 Federated provisioning overview

Provisioning requests sent between identity providers and service providers must be secure and be based on open standards. A standard that satisfies these requirements is WS-Provisioning. See Figure 2-13. These requirements may be satisfied by 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. See 2.3.8, “WS-Provisioning” on page 57, for more details on the WS-Provisioning standard.

WS-Provisioning 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.

2.7 On demand security reference architecture

Some initial work has been done to define a comprehensive *on demand Security Reference Architecture*; see Figure 2-14 on page 73, which illustrates this work.

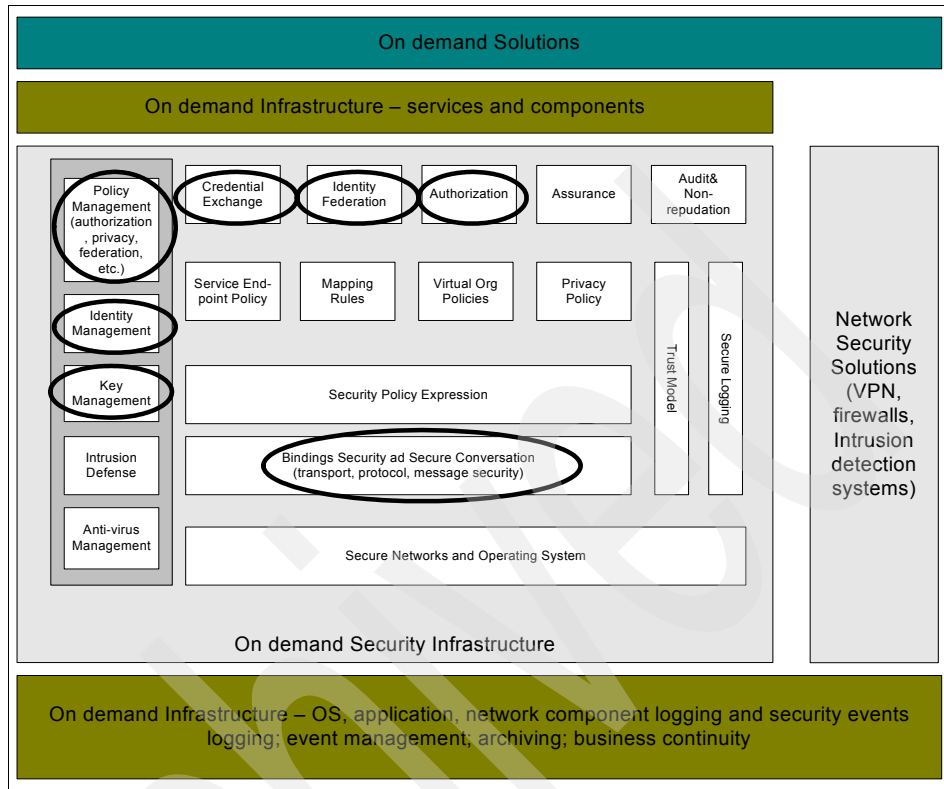


Figure 2-14 On demand Security Architecture (logical)

A complete discussion of this architecture is beyond the scope of this book, which focuses specifically on the area of federated identity management. The components that relate FIM are circled in the diagram in Figure 2-14. We give a short overview of each of them in this section. A brief description of the other components is included in the glossary of this document.

2.7.1 Policy management

Policy management in the *on demand Security Infrastructure* is the consistent application of enterprise security policy to on demand infrastructure components, services, and applications; Network Security Solutions; and on demand Security Infrastructure components and services. Policy management is applied independent of application logic and operating system platform, and includes trusted identity and token life cycle management identity, access control/authorization life cycle management, federated identity life cycle, privacy, single sign-on, compliance determination and re-mediation, security event auditing and processing, and failure situations.

2.7.2 Identity management

In accordance with document security policy, identity management includes the following:

- ▶ Identity proofing, identity approval, and identity rights authorization
- ▶ Identity token creation and token distribution to the user
- ▶ (Dynamically) provisioning user identity, rights, and profile to relying parties (operating systems and applications)
- ▶ User profile management
- ▶ Enabling user self-care
- ▶ Delegating administrative responsibility for approval and authorization as needed
- ▶ Processes for token changes IAW policy, revoking, and approving reissue of new/changed token
- ▶ Performing identity management in accordance with security policy

In the context of identity management we have to use the following definition to clearly distinguish between token and credentials:

Token

An object(s) that an entity possesses and controls (typically a key or password) used to authenticate the entity's identity. The token is provided to the entity as a result of successfully completing the identity proofing and registration processes.

Credentials

Objects used in authentication that bind an identity or an attribute to a subscriber's token. Note that this document distinguishes between credentials and tokens.

2.7.3 Key management

In accordance with document policy, key management provides life cycle management for public-private key pairs using a trusted Public Key Infrastructure (enterprise or outsourced) operating in accordance with a documented certificate policy. Private keys and X.509 certificates can be used to provide authentication, confidentiality, data integrity, and non-repudiation for transactions and other data.

2.7.4 Credential exchange

The purpose of a credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across

networks and among the platforms, the processes, and the security subsystems within a computing solution. Credentials are created as a result of a successful authentication. Some common types of credentials are:

- ▶ X.509 public key identity certificates that bind an identity to a public key
- ▶ X.509 attribute certificates that bind an identity or a public key with some attribute
- ▶ Kerberos tickets that are encrypted messages binding the holder with some attribute or privilege
- ▶ Encrypted cookies

Credential exchange is the process of passing a credential from one entity to another entity using a protocol trusted by both entities or a protocol in which both parties can establish mutual trust.

2.7.5 Identity federation

Identity federation is the life cycle management of cross-enterprise identities. Such identities may be centrally managed or rely on trusted third parties. Trust federation includes:

- ▶ Trust management
- ▶ Trust brokering
- ▶ Single sign-on
- ▶ Cross-enterprise identity mapping
- ▶ Cross-enterprise identity provisioning

2.7.6 Authorization

Authorization, also called rights and permissions, is allowing only users that are approved to access and receive the benefit of systems, data, applications, and networks (public and private). Authorization management is a life cycle process for authorization data.

2.8 On demand integration reference architecture

The IBM WebSphere Integration Reference Architecture, as represented by this high level, logical reference architecture, is a middle ware platform that provides elements for function isolation. This modular middle ware platform provides the development and operating environments for SOA-based solutions. This chapter will give a brief overview of the layers involved in this reference architecture represented by the Figure 2-15 on page 76. and how key parts relate to federated identity management.

SOA is going to be a major driver in the Web services security management space since one of the aspects of SOA is the ability to expose enterprise services outside of the enterprise in a standardized way. To accomplish the inter-enterprise integration there will be a heavy requirement on having Web services security management in place to manage the application and-to-end security.

So to understand a little better the components involved lets take a look at the IBM WebSphere Integration Reference Architecture.

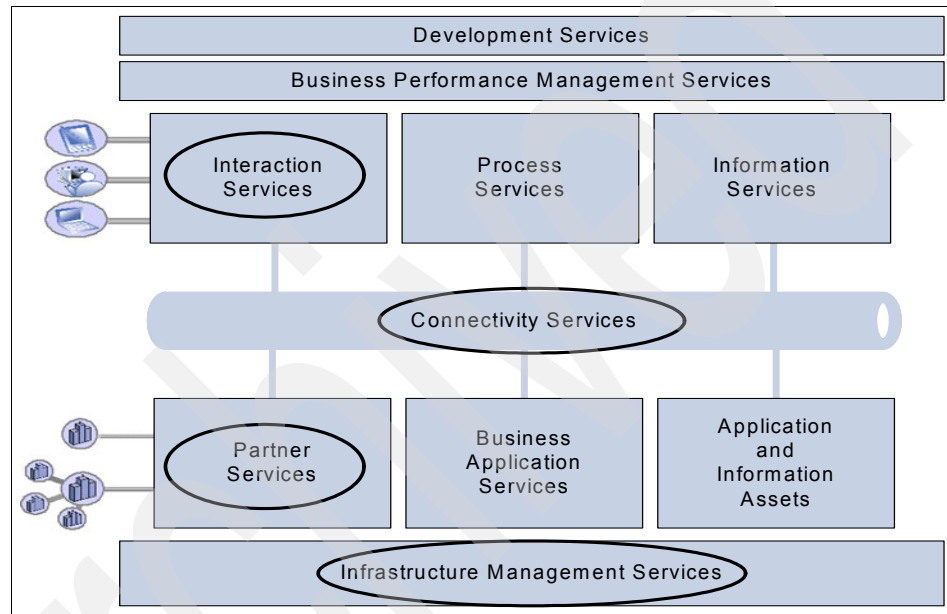


Figure 2-15 IBM WebSphere Integration Reference Architecture

A complete discussion of this architecture is beyond the scope of this book, which focuses specifically on the area of federated identity management. The components that relate to FIM are circled in the diagram in Figure 2-15. We give a short overview of each of them in this section.

In the center of the architecture is a set of Connectivity Services. These services provide the most fundamental of functions required for any integration infrastructure, the ability to inter-connect multiple services spread throughout the enterprise in a fully distributed implementation. For an in-depth understanding of this core element see the IBM Redbook *Patterns: Implementing an SOA using an Enterprise Service Bus*, SG24-6346-00.

Just below the Connectivity Services are a set of elements where business services run. First, existing application and information functions must be accessed through a set of Access Services in the Application and Information Assets element. New business services run in another element and leverage its Business Application Services. Services provided through relationships with third party partners and suppliers are integrated into the architecture through the Partner Services element.

Just above the Connectivity Services are a set of elements that facilitate the integration of people, processes, and information.

Business Performance Management Services capabilities are provided across all these elements facilitating business and IT level monitoring and management. Development Services provide the role-based, model driven development platform required for the efficient development of solutions.

All the layer are built on a set of Infrastructure Management capabilities that allow it to be used to deliver any level of quality of service for any enterprise level requirements. This is where federated identity management has many of its capabilities.

2.8.1 Connectivity services

At the core of the Reference Architecture is a set of Connectivity Services. This architectural element delivers all the inter-connectivity capabilities required to leverage and use services implemented across the entire architecture. Transport services, mediation services, and event services are all provided.

2.8.2 User interaction services

The user interaction services elements are set of services that are oriented toward the integration of people, processes, and information. These services control the flow of interactions and data among people and automated application services in ways appropriate to the realization of a business process. User Interaction Services provide the capabilities required to deliver IT functions and data to end users, meeting the end-user's specific usage preferences. Capabilities important to the integration of specific devices such as sensors and actuators used on remote equipment is also supported.

The user interaction services would be the portal where Web based services would be presented. The user interactions discussed in 2.4, "Federated single sign-on" on page 59, would be to this service. Federated single sign-on (F-SSO) would be leveraged if federation would be required between inter enterprise portals.

2.8.3 Application and information assets

Existing enterprise applications and enterprise data are accessible from the Connectivity Services through a set of Access Services. These access services provide the bridging capabilities between legacy applications, pre-packaged applications, enterprise data stores (including relational, hierarchical and nontraditional, unstructured sources such as XML and Text), and so on, and the Connectivity Services. Using a consistent approach, these access services expose the data and functions of the existing enterprise applications, allowing them to be fully re-used and incorporated into functional flows that represent business processes. Existing enterprise applications and data leverage the functions of their operating environments such as CICS®, IMS™, DB2®, and so on. As these applications and data implementations evolve to become more flexible participants in business processes, enhanced capabilities of their underlying operating environments, for example support of emerging standards, can be fully utilized.

User attributes, as discussed in 2.2.5, “Identity attributes” on page 45, in the applications and information assets that needed to be shared outside the enterprise, would be managed by an enterprise identity management solution and federated provisioning solution to integrate with customers, partners and suppliers who required the information.

2.8.4 Business application services

The business application services element contains a set of services that provide runtime services required for new application components to be included in the integrated system. These application components provide new business logic required to adapt existing business processes to meet changing competitive and customer demands of the enterprise. Design and implementation of new business logic components for integration enables them to be fully re-usable, allowing them to participate in new and updated business processes over time. The business application services include functions important to the traditional programmer for building maintainable, flexible, and re-usable business logic components.

2.8.5 Partner services

In many enterprise scenarios, business processes involve interactions with outside partners and suppliers. Integrating the systems of the partners and suppliers with those of the enterprise improves efficiency of the overall value chain. Partner Services provide the document, protocol, and partner management services required for efficient implementation of business-to-business processes and inter-actions. To support this, a set of services must be available in the infrastructure services element, handling end to

end federation and security. Partner services is the consumer of FIM/Web services security management services, which are located in the infrastructure services.

2.8.6 Infrastructure services

Underlying all these capabilities of the WebSphere Integration Reference Architecture is a set of infrastructure services which provide security, directory, IT system management, and virtualization functions. The security and directory services include functions involving authentication and authorizations required for implementing, for example, single sign-on capabilities across a distributed and heterogeneous system.

IT system management and virtualization services include functions that relate to scale and performance, for example edge services and clustering services, and the virtualization capabilities allow efficient use of computing resources based on load patterns, and so on.

While many infrastructure services perform functions tied directly to hardware or system implementations, others provide functions that interact directly with integration services provided in other elements of the architecture through the Connectivity Services. These interactions typically involve services related to security, directory, and I/T operational systems management.

2.9 Method for architecting secure solution

Addressing an overall enterprise security architecture is a somewhat complex undertaking because it involves many areas, identity federation being one of them. IBM has introduced a new methodology called Method for Architecting Secure Solutions (MASS) that will be used by IBM Global Service employees in future security architecture engagements. It helps understand and categorize security-related problems and discussions in today's e-business-driven enterprise IT infrastructures. This discussion was originally posted in a special edition of the *IBM Systems Journal on End-to-End Security*, Vol. 40, No. 31.

The task of developing IT solutions that consistently and effectively apply security principles has many challenges, including the complexity of integrating the specified security functions within the several underlying component architectures found in computing systems, the difficulty in developing a comprehensive set of baseline requirements for security, and a lack of widely accepted security design methods. With the formalization of security evaluation criteria into an international standard known as Common Criteria¹, one of the barriers to a common approach for developing extensible IT security architectures has been lowered; however, more work remains. The redbook

Enterprise Security Architecture using Tivoli Security Solutions, SG24-6014, describes a systematic approach for defining, modeling, and documenting security functions within a structured design process in order to facilitate greater trust in the operation of resulting IT solutions.

There is also another redbook *Identity Management Design Guide with IBM Tivoli Identity Manager, SG24-6996*, which relates very closely to the topic of this book. It discusses the identity management of individual users within an enterprise and its systems. The redbook describes the aspects of the architecture and design of an identity management solution, and how to implement the user life cycle management including the provisioning of the user information with all the applications and systems in the enterprise. This redbook extends the identity management discussion to federated identity management.

2.9.1 Implementation flow

Most projects will involve business, project management, and technical tasks. The steps are discussed below.

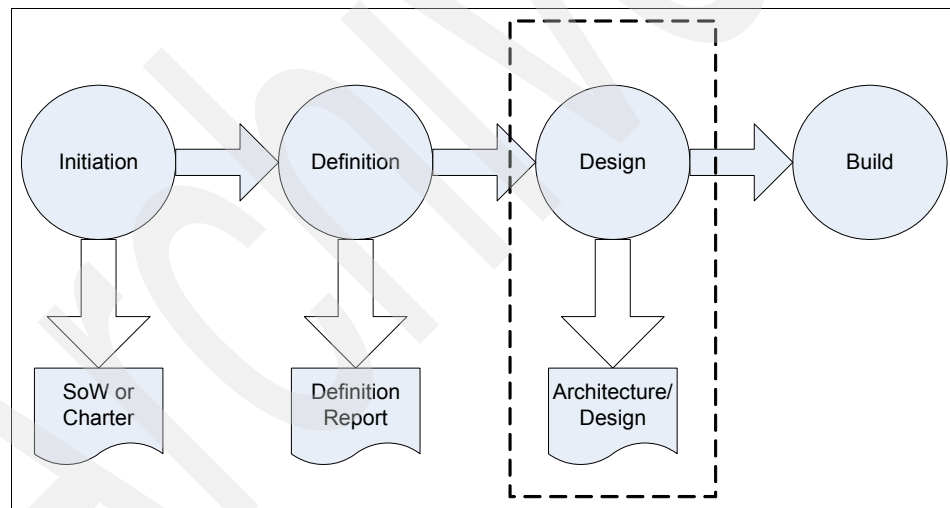


Figure 2-16 Generic implementation phases for a project

The steps in Figure 2-16 are described below:

Initiation This is the project initiation step. It will normally involve identifying the project background and requirements at a

¹ This is a set of tests originally based on the US Orange book and European/Australian ITSEC evaluations. It is currently recognized by 14 countries. There are seven levels of tests. Evaluation Assurance Levels (EAL) 1–4 are usually used in the commercial areas, while the tests representing the higher EALs 5–7 are reserved for the security testing of highly secure environments.

high level. The deliverable for this step will be some sort of Statement of Work (SoW) or Project Charter. The high-level requirements will have come from a preceding project (such as an IT architecture or security architecture project) or the software purchase requirements.

Definition

This is the project definition step, for example, where the project is defined in detail, sometimes referred to as the solution outline. This involves gathering the details: The existing systems, users, procedures, and other information, and the detailed requirements of the solution. The deliverable for this step will be one or more documents defining the project. These may include a Project Definition Report, a Requirements document, a Functional Specification, and an Existing System Analysis document.

Design

This is the design step. It involves designing the solution, or a macro design. The deliverables for this phase are the Architectural (at least Architecture overview, operational model and non-functional requirements) and Design (at least use cases, and component model) documents.

Build

This is where the solution is built and implemented. From the macro design in the previous step micro designs are the base for the build. There will be a micro design for each release. The micro designs are governed by change cases which feed back to the macro design as well.

As mentioned, the focus of this redbook is on the design phase and production of the Architectural and Design documents (as highlighted in Figure 2-16 on page 80). However, much of the information required for the design will have been gathered and documented in the Definition phase. The next section looks at this.

2.9.2 Definition phase of a federated identity management solution

The definition phase defines the project in detail, and involves detailing the current environment, the problem to be solved by the solution, and the detailed requirements for the solution.

The initial project definition will be based on the documentation that triggered this project, such as the IT Architecture, Security Architecture, INtegration Architecture, RFP, or equivalent. These documents identify the business background; the business need for the solution; and, normally, the business and technical requirements for the solution.

For a federated identity management solution, the following areas need to be defined in this phase (in no particular order). This list assumes that we are creating a federation between two business partners. If there are several federation business partners most of the steps needs to be worked out per business partner:

- ▶ Selection of the federation protocol profile(s): The definition of the federation protocol profiles to be used between the two business partners.
- ▶ The roles of identity and service provider: The definition who is the authoritative source of the user identity information.
- ▶ Identity/attribute mapping: The definition of the attributes to be shared and the mapping of them in the business partner systems.
- ▶ 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/de-provisioning procedures.
- ▶ User account management/provisioning: The procedures for managing user identity data, agree what information can be shared, and what information is independently managed by users, and will the users be provisioned automatically to the new endpoint (a priori or runtime).
- ▶ Security policy: What the corporate security policy defines for users, accounts, passwords, and access control, and how they will be affected with the federation.
- ▶ Interfaces: The interfaces to the current identity management mechanisms and procedures and the integration requirements of the solution.
- ▶ Auditing and reporting procedures: The procedures for auditing and reporting, who is involved in the auditing and reporting of users and their access, the audit requirements for the solution, and the reporting requirements for the solution.
- ▶ Technical requirements: The other technical requirements for the solution, such as availability and recovery.

The first five bullets are federation-related items, and the sections later in this chapter give more detailed information on these topics. The other bullets are more general security-related items, which have been discussed in more detailed in the other redbooks referenced in the beginning of this chapter.

Gathering this information normally involves a series of interviews and workshops with the relevant people from the business partner organizations involved in the federated identity management project. The combination of these interviews and workshops will develop a picture of how the system currently works and how it could be improved with the federation. It is important to

evaluate the wish list from the genuine requirements. The project owners should drive the requirements for the proposed system, although others may contribute to an understanding of the need for the requirements.

A key component of delineating the definition and design phases is that the existing system and solution requirements are agreed between the project owner and the project team prior to the commencement of the design phase.

2.10 Conclusion

In this chapter we have discussed the architecture and design of a federated identity management solution between trusted business partners. In the beginning we stated that, in general, building a particular design is just one part of an overall implementation of a certain solution. The whole project consists of a number of steps, starting from the definition of the business context, gathering the requirements (both the functional and non-functional), creating the architectural design, and finally building the solution. This chapter focused on the architectural design aspect of an overall project.

In order to help our customers to build a FIM solution, IBM has created a methodology for building a security solution, including the architecture and design, and which is used by IBM Global Services employees in security architecture engagements.

We also discussed some of the architectural considerations when building a FIM solution. We discussed some of the FIM-specific functionality to give a better understanding to the reader of the federation-related features like single sign-on, account linking, single sign-off, protocol profiles, provisioning, and so on.

At the end of the chapter we described the status FIM standards and interoperability at the time of writing this redbook, which also shows that FIM solutions can be implemented today.

Archived



Tivoli Federated Identity Manager architecture

The previous chapter described an overview of the capabilities of a general federated identity management solution. These capabilities are treated as individual logical functions that may be leveraged in a FIM solution. The capabilities are logical in that they are not implemented by one-to-one corresponding functional components. Instead, federation functionality is provided by a set of services that are composable in order to create the functional capabilities described earlier.

In this chapter we introduce the high-level components and new concepts for the design of a federated identity management solution using IBM software technology.

This chapter provides you with an understanding of the following topics:

- ▶ The high-level logical services architecture for IBM Tivoli Federated Identity Management
- ▶ A more detailed look into federated single sign-on (F-SSO), Web services security management, and provisioning solutions

In this chapter we refer to various IBM offerings. More detailed descriptions of them can be found in Chapter 5, “Integrating with IBM identity management offerings” on page 171.

3.1 Federated Identity Management functionality

The FIM functionality is built around a trust infrastructure implemented by the Tivoli Federated Identity Manager trust service. This infrastructure is the basis for the Tivoli Federated Identity Manager solutions provided for federated provisioning, federated single sign-on, and Web services security management. Each of these solutions may be deployed independently of the other. Likewise, they can all be deployed in the same environment to provide a complete federation solution.

As shown in Figure 3-1, Tivoli Federated Identity Manager provides overall functionality for:

- ▶ Identity federation
- ▶ Federated provisioning
- ▶ Web single sign-on
- ▶ Web services security

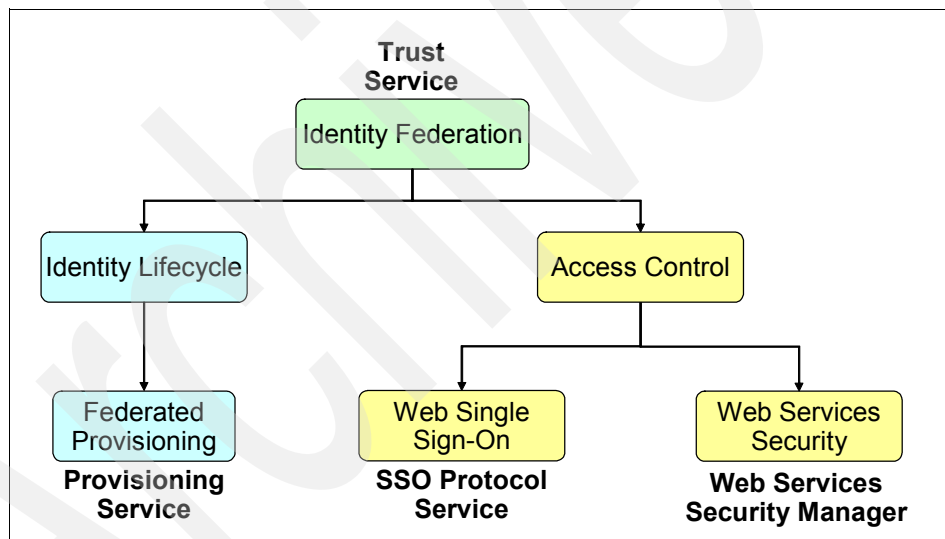


Figure 3-1 Tivoli Federated Identity Manager runtime services

Federated Identity Management service components are described in 3.2, “Federation services” on page 87. These components represent individual services that may exist as distinct services or as logical services within TFIM. Logically, each of these functional components is represented by a logical service, so that:

- ▶ Federated provisioning functionality is provided by the provisioning service
- ▶ Web single sign-on is provided by the single sign-on protocol service
- ▶ Web services security is provided by the trust service

Note that the Web services security management functionality directly leverages the trust service. The single sign-on protocol service (SPS) in turn leverages the trust service as an *internal* SPS service. The provisioning service (PS) may or may not leverage the trust service, based on the requirement to security (via Web services security management) the provisioning requests.

3.2 Federation services

Tivoli Federated Identity Manager services facilitates a standardized means for allowing businesses to:

- ▶ 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 referring to a Principal, their 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.

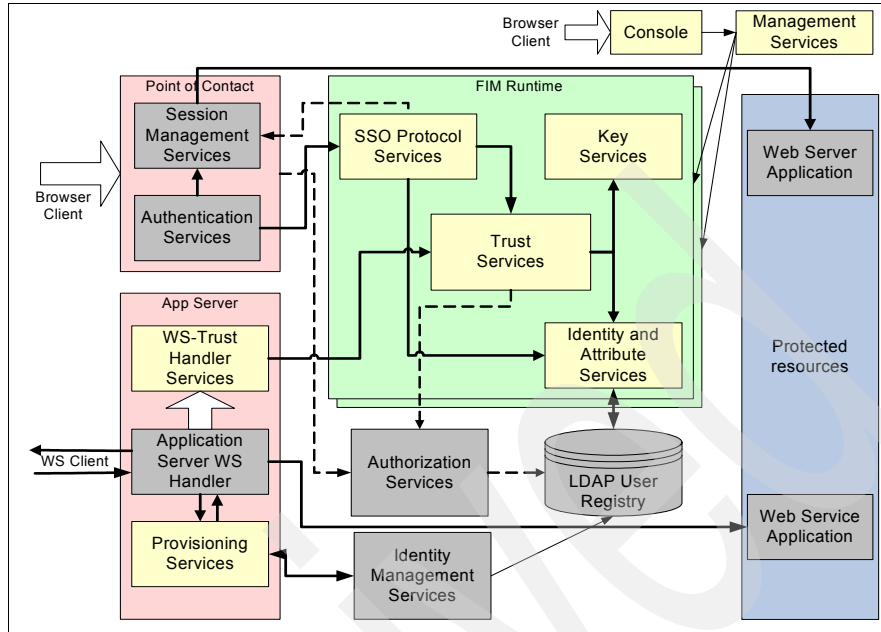


Figure 3-2 FIM services architecture - The full picture

The following sections give an overview of each of the services components represented in Figure 3-2, 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-grey boxes 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 3-3 on page 89, 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, since they differ in layering and protocols.

In the application-based interaction to the left in Figure 3-3 on page 89 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.

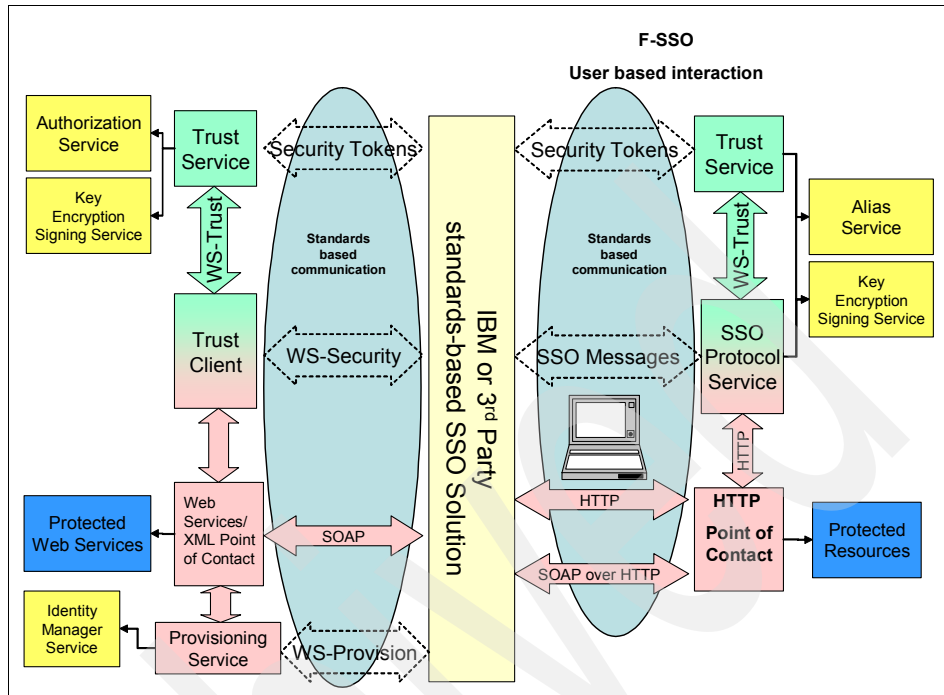


Figure 3-3 User and application-based interaction components and their communication

3.2.1 Point of contact (PoC)

The point of contact is used for HTTP-based user interactions. The point of contact 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 end user (or a user agent such as the browser) and the authentication service.

Within a federated single sign-on environment, the challenge/response protocol is not always negotiated with the end user but may be negotiated with a third party acting on behalf of the user. This third party will usually assert some form of security token or assertion about the user based in its own (local) authentication of the end 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 single sign-on service is used. SSO services (described in 3.2.2, "Single sign-on protocol services (SPS)" on page 91) 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 will generate 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 3.2.6, "Authorization services" on page 97.

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.

3.2.2 Single sign-on protocol services (SPS)

Within a federation environment, federated identity management protocols are used to communicate information about a user between federation business partners. For example, with federated single sign-on 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.

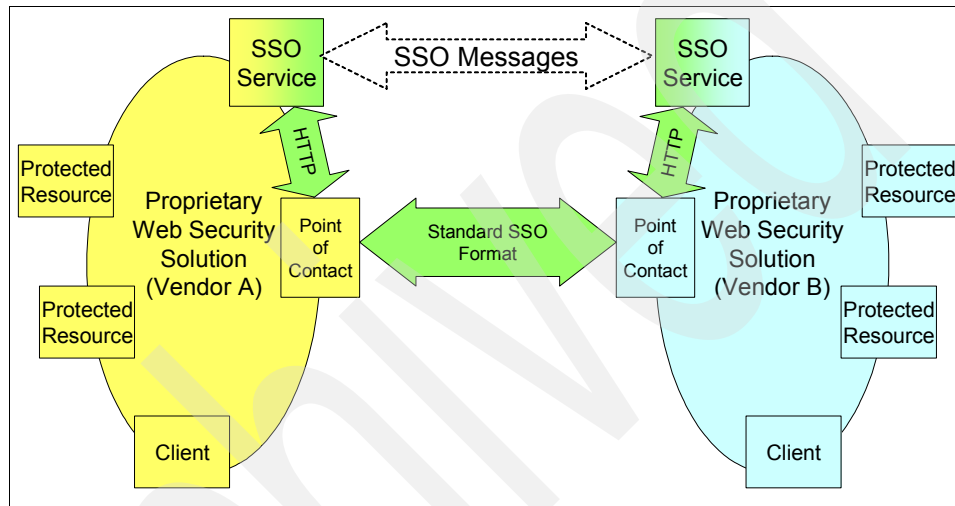


Figure 3-4 Externalized SSO Services

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 3-4. Single Sign-on protocol endpoints are instead hosted by a separate service, the single sign-on protocol service. The point of contact server still maintains control of user sessions, providing session management services.

The point of contact server has a number of interfaces to the SSO Protocol Service but these do not need to be modified in order to support different (or new) single sign-on standards. Only the SSO Protocol Service has to be modified if changes to single sign-on behavior are needed.

External Authentication Interface (EAI)

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 point of contact, the federated single sign-on

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. See Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, for a detailed description.

3.2.3 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 partners' 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 it 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.

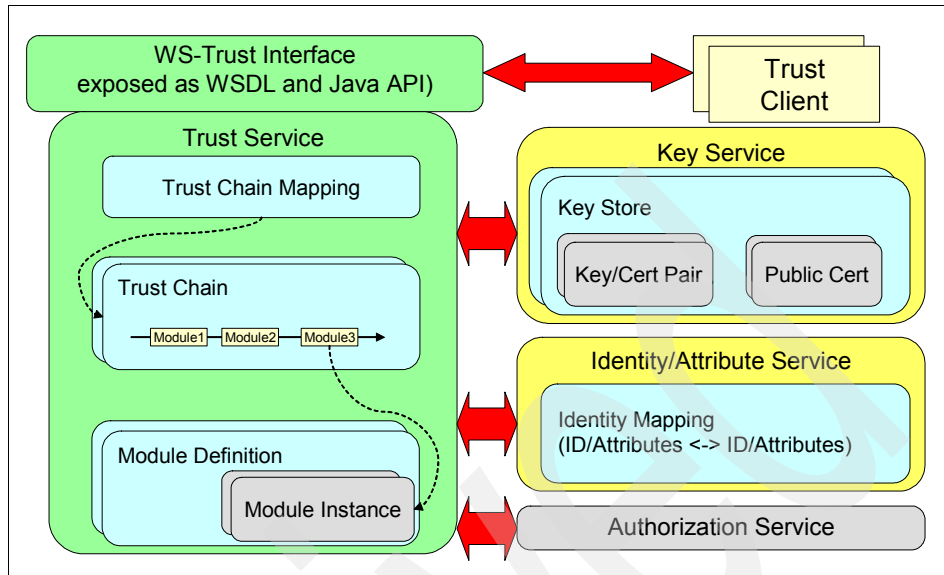


Figure 3-5 Trust service components and connections

Figure 3-5 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.

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, it is not enough to simply understand the different token standards. It is just as important to know 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. It is even possible 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. There are two reasons for having this abstract format:

- ▶ First, 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 it is possible to convert from any token to any other token via the abstract format.
- ▶ Second, to be able to perform identity mapping. This mapping is made much simpler if the mapping module only has to deal with 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.

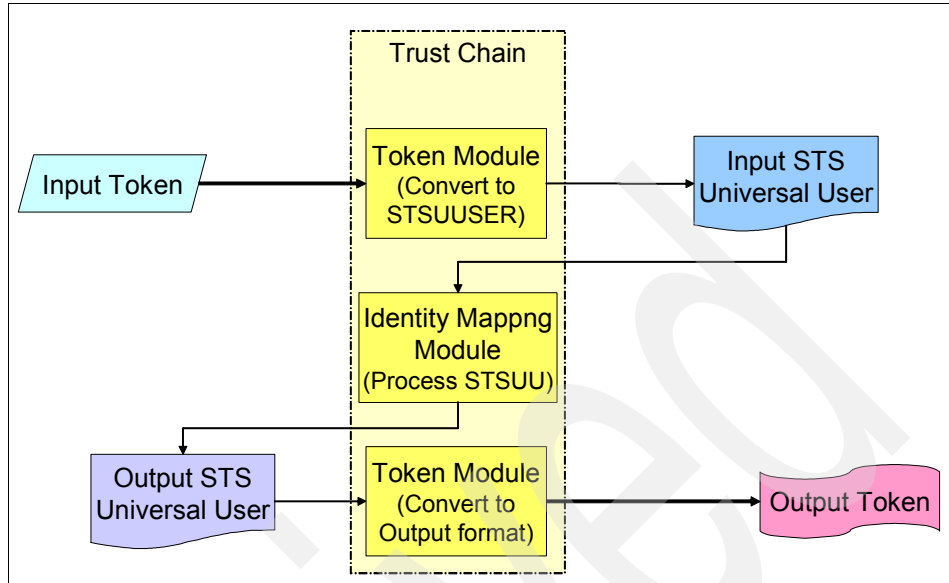


Figure 3-6 Trust service processing for federated single sign-on

Figure 3-6 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.

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 creates an XML document with known structure. All of the 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.

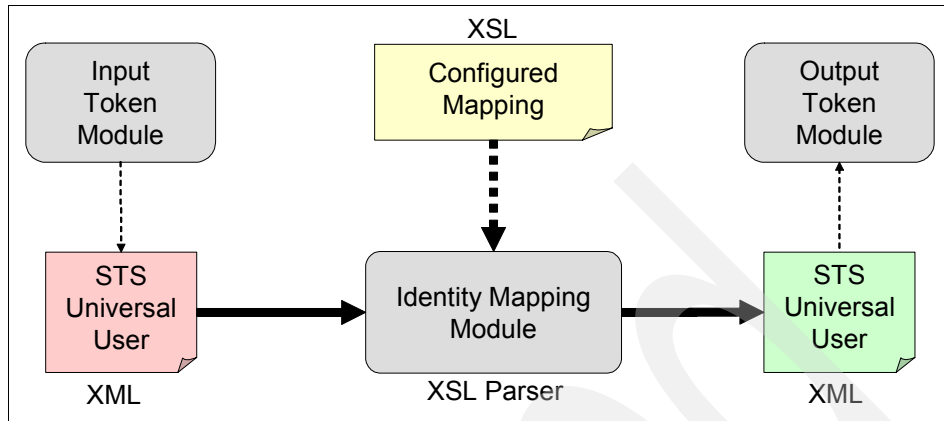


Figure 3-7 Trust service transformation engine

Figure 3-7 shows how the Identity Mapping module is implemented using an XSL parser.

The input STSUUSER document is generated by the input token module. This is an XML document. 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 take both of these things into account. More details on these requirements are covered in Appendix B, “Identity mapping rules” on page 381.

3.2.4 Key services (KESS)

Key services are leveraged to provide access to key stores used by a trust service and the SSO Protocol Service. This allows the trust service and SPS to plug in/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.

In Appendix C, “Keys and certificates” on page 425, there is a detailed description of key and certificate generation related to Tivoli Federated Identity Manager and specifically the use cases in Part 2, “Customer environment” on page 181, of this book.

3.2.5 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 some backing data store.

Identity services are leveraged by many different services within a federation environment. The authentication service will leverage 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 TFIM 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 and Attribute Service (IdAS)*, provides the functionality required to manage the attributes required for a security token.

An IdAS will normally access 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 may leverage a private, internal data store.

3.2.6 Authorization services

Authorization services are responsible for providing access decision point functionality within a security model. The authorization service itself may not act as an *access enforcement function (AEF)*. AEF functionality is typically provided

by Session Management Services. Tivoli Access Manager provides AEF functionality with TAM WebSEAL acting as an ADP.

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 may respond with a simple yes/no, indicating whether an access request is allowed. Based on this response, session management services act as the authorization enforcement point by allowing/disallowing the actual request for access.

3.2.7 Provisioning services

Provisioning services are used within a federated environment for both a priori and run-time provisioning solutions. Provisioning services interact with both local identity management systems (such as Tivoli Identity Manager) and local data stores (access via 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.

Provisioning services are leveraged as part of the identity management functionality within an enterprise; as such, they are often integrated with a local identity management (IM) system. This allows a local IM to treat a federation business partner as a local provisioning endpoint, including this endpoint in any workflow-based approval processes that are in place. A local IM can then provision information about a user to a federation business partner, including provisioning changes to a user's personal profile (for example, home address), status (for example, on leave of absence), or subscriptions (for example, signed up for corporate-sponsored cell phone service). This allows an identity provider to have a seamless and consistent view of managing a user across a federation while allowing federation business partners to benefit from the management functionality assumed by the identity provider.

3.2.8 Management Services

The management services are used for Tivoli Federated Identity Manager runtime configuration and deployment. The interfaces to the management services are:

- ▶ ISC - The new IBM Integrated Solutions Console providing a single portal style administrative console for Tivoli Federated Identity Manager.
- ▶ API - Used by, for example, the InfoService; see 3.3.6, "InfoService" on page 119.

This combination of API and (Web-based) management console provides management flexibility and allows a customer to tailor management experience as appropriate.

Console

Tivoli Federated Identity Manager uses a new console framework called the IBM Integrated Solutions Console (ISC). Many IBM products are moving to use this framework with the aim of providing a single portal style administrative console that can be used to manage multiple IBM products from one place.

The ISC is based on cut-down versions of WebSphere Application Server 5.1 and WebSphere Portal Server. All of this is installed as part of the installation of the ISC. Since Tivoli Federated Identity Manager components require WebSphere Application Server 6.0, the ISC cannot share the same WebSphere Application Server instance as Tivoli Federated Identity Manager components. However, WebSphere 6.0 and the ISC can be installed on the same machine without conflict.

Once the ISC is installed, console plug-ins are deployed into the ISC; see Figure 3-8. The Tivoli Federated Identity Manager Console is one such plug-in. The ISC is accessed over HTTP(S). This means that the Tivoli Federated Identity Manager administration console can be accessed from any client that has connectivity to the machine where the ISC is installed.

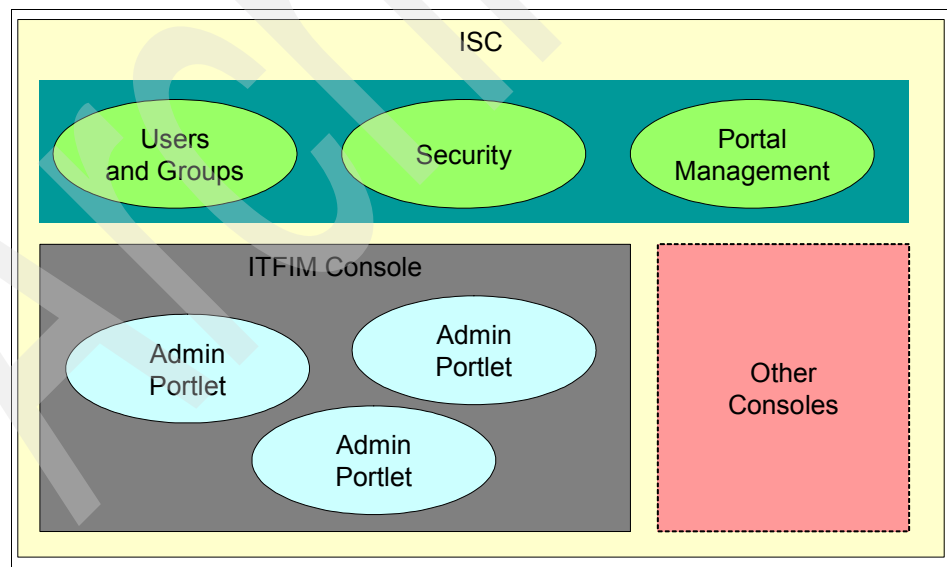


Figure 3-8 Tivoli Federated Identity Manager (ITFIM) Console within the ISC

Deployment manager

The ISC console interface uses the deployment manager to push deployment and configuration to remote Tivoli Federated Identity Manager nodes, as shown in Figure 3-9. The deployment manager supports multiple domains and clustered nodes (more on clustered nodes in 4.1.5, “Highly available architecture patterns” on page 147). WebSphere Application Server functionality used to synchronize the configuration files to clusters and the Tivoli Federated Identity Manager Runtimes on the WebSphere Application Servers read the files locally.

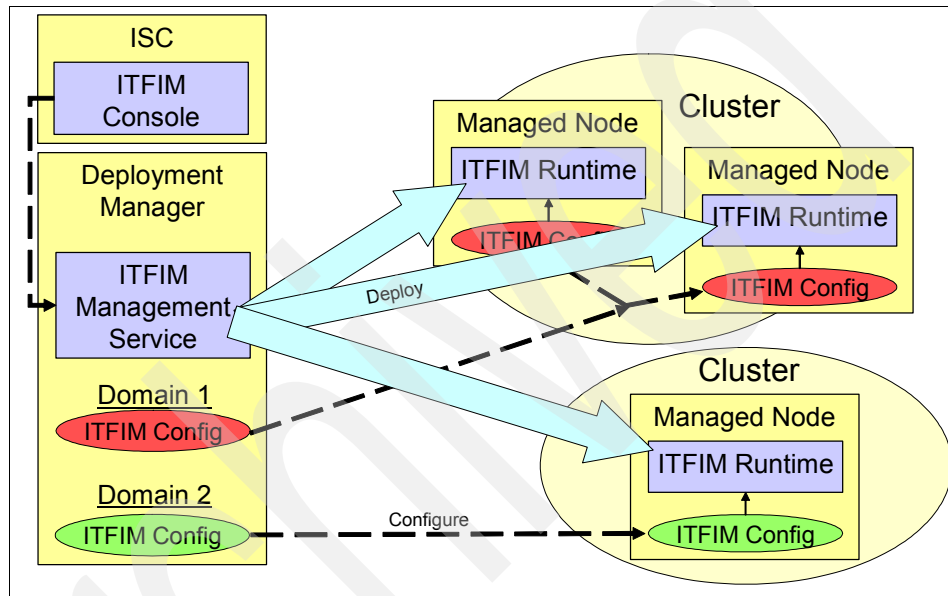


Figure 3-9 Tivoli Federated Identity Manager Deployment and configuration of Tivoli Federated Identity Manager and Tivoli Federated Identity Manager clusters

3.3 Federated single sign-on

Federated single sign-on 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/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, there are two main areas where participants must agree on the technology choice in order to achieve interoperability.

The first area is 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 3.2.3, “Trust services” on page 92.

The second area is the single sign-on protocol. This 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. These areas are discussed more later in this chapter

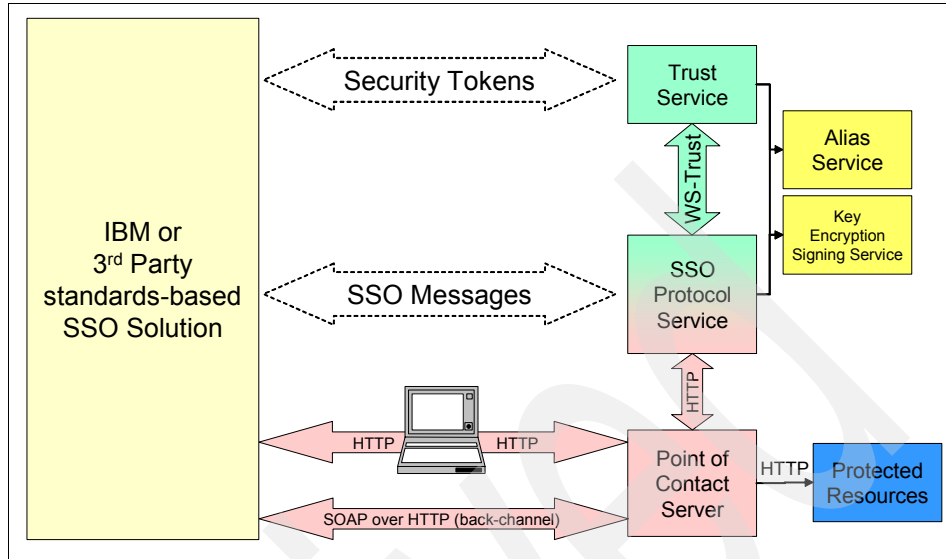


Figure 3-10 Single sign-on components and communication

Figure 3-10 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).

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 via 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 via 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 via the SSO Protocol Service.

3.3.1 Architecture overview

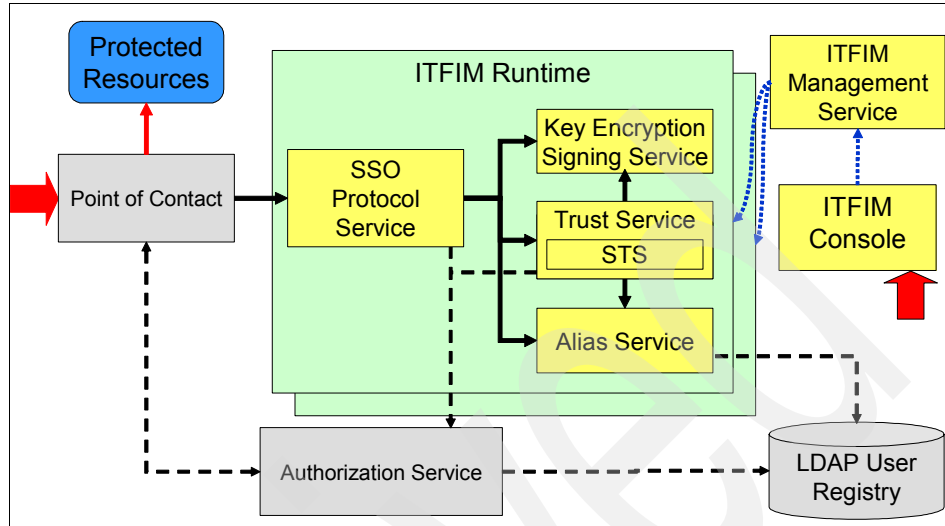


Figure 3-11 Tivoli Federated Identity Manager components for Federated SSO

Figure 3-11 shows the Tivoli Federated Identity Manager architecture required to support Web-based single sign-on protocols such as Liberty, WS-Federation, and SAML 1.0.

All outside communication with the environment comes via 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 single sign-on processes itself.

The SPS communicates with the trust infrastructure components in order to build and consume SSO messages and uses the Access Manager Admin 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.

In the following chapter the different types of F-SSO protocol functionality are covered.

3.3.2 Trust in F-SSO

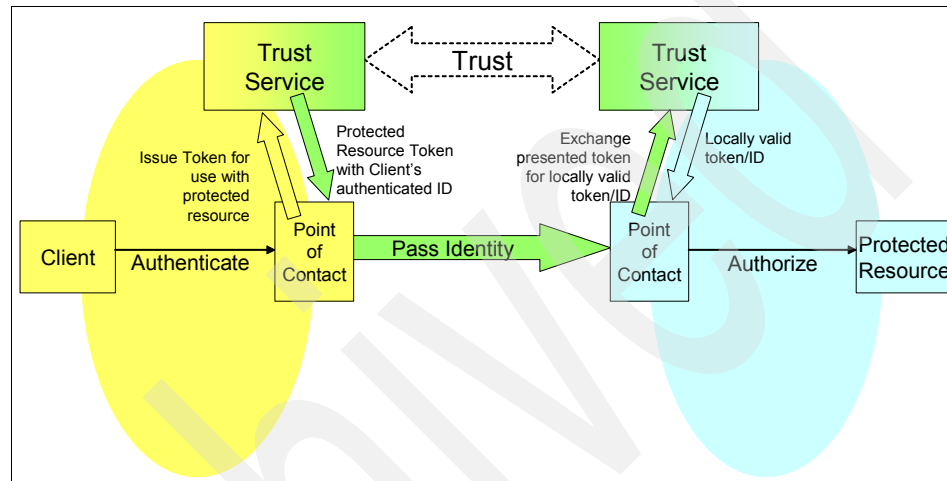


Figure 3-12 Using trust service in F-SSO

Security tokens are included in a message to pass an identity and to convey security-specific information (used for authentication and/or authorization purposes, for example) about a requestor; see Figure 3-12. 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/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.

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 allows the trust service to manage a business partner's token meta-data together with the business partner's cryptographic material.

3.3.3 F-SSO protocol functionality

Tivoli Federated Identity Manager and Access Manager for e-business together provide support for browser-based federated single sign-on protocols (F-SSO). 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 federated single sign-on protocols.

Single sign-on (SSO)

Single sign-on is a well-understood process. This is the process of allowing a user, authenticated to one domain (their *home domain* in Access Manager terms, also known as their 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 (as all other CD-SSO protocols) come in two flavors: *Push* and *pull*.

Push protocol

In push protocol the user invokes a remote resource from within the control of their home domain (through a link on a portal page, for example), and is redirected to the remote resource, carrying their vouch-for token with their request. This means that the service provider (site of the remote resource) does not need to prompt the user for information about their 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; 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). As 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 their 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).

Once 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 will allow the service provider to implement local access control policies, for example, for the user's session.

Single logout (SLO)

Previous attempts at single sign-on have often neglected the corresponding single sign-off 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 is a strong recommendation/requirement that stems in large part from the liability normally assumed by an identity provider for a user within a F-SSO relationship.

It is not always the case that logout should be an allowable service provider action. This follows in that if a user has single signed-on to the service provider, he may well have no notion that he has 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 time-out) at a service provider and rely on single sign-on to re-establish a session at a service provider, perhaps many times within the lifetime of the user's identity provider session.

If a user is presented with a global logout option at the service provider, this 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 that logout in general has implications for things such as session duration (differing durations at identity providers and service providers). In general, the inactivity time-out set for an identity provider should be longer than that set for its service provider business partners. This will prevent 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 run-time 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 desires 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 need to have some 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, there will be a write operation to an identity store to allow the saving of the linking/mapping information.

In some cases, the account linking process will set 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 will 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 if/when a user de-federates the accounts, she may still access her 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 is *not* provisioning for two important reasons: One, it requires that the user already has pre-existing accounts at both the identity and service provider. Two, 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, as described in 3.5, "Provisioning services" on page 129. 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.

Password synchronization

Password synchronization may be a requirement for some relationships that entail both federated user life cycle and Web services provisioning management solutions. As password synchronization may require provisioning functionality, it is also discussed in the Web services provisioning section.

With F-SSO, a service provider may be reluctant or unable to turn off direct access to their resources, meaning that they must allow a user to authenticate to the service provider as well as gain access as the result of federated SSO. In order to achieve the benefits of federation (which often revolve around the cost of password management and password reset), some companies will synchronize passwords across participants. This at least will allow the service providers to rely on the identity provider for password management, including Help Desk calls. It will also simplify password management for the user, as it has the same effect as a user-enforced global password. Note that password synchronization is not as simple of a solution to implement, as differing password management policies must be taken into account.

We expect that password synchronization solutions will not be common. What is more likely is that a service provider will disable the password at the service provider side once account linkage has been accomplished (without disabling the user's account). This means that the user can only access the service provider resources from their identity provider. If/when account de-linking (see the next section) occurs, user self-care can be invoked to allow the user to re-establish a password for local access.

Account de-linking (name de-federation)

Just as account linking is the process of establishing a linking, or mapping, between a user's accounts across federations, account de-linking is the process of removing any reference to or knowledge of that mapping.

Account de-linking may occur in a B2C scenario when a user changes his identity provider (moving from Internet service provider A to Internet service provider B, and therefore forcing a change of identity provider, for example), or when a user changes service providers (changing his bank account from bank A to bank B).

Account de-linking may occur in a B2B2E scenario when an employer changes service providers (moving from benefits A to benefits B as medical benefits providers, for example), or when a user changes employers (moving from company A to company B but keeping his account with pension fund A for retirement fund purposes).

Account de-linking may be triggered at the identity provider (for scenarios where the user is changing service providers or simply wishes to remove F-SSO

functionality between the IdP and SP) or at the service provider (when the user wishes to establish a new IdP or wishes to remove F-SSO functionality at that SP).

Note that account de-linking is a single step and does not require/force a user to establish a new account linking relationship.

Where are you from (WAYF)

Where are you from is the process of determining (by a service provider) where a user's home domain (or identity provider) is located. Where are you from has two profiles: *Active* and *passive*.

With a passive WAYF, the service provider has already established some form of (long-term) information that it can access to determine a user's identity provider. This simplest *form* of WAYF information is configured into the URLs associated with single sign-on, so that a request for single sign-on received at <http://www.fabrikam.com/fim/idpAsso.html> is always associated with IdP A.

A more likely form of storing WAYF information is in the form of a domain cookie that identifies the user's identity provider and nothing else. There is no security-relevant information of any form stored in this cookie. If a user attempts to access a service provider resource and is not carrying some form of F-SSO token, the service provider will look for a WAYF cookie to determine the user's home domain. Based on the identity provider information stored in this cookie, the service provider will be able to determine (based on local configuration) the corresponding F-SSO endpoint at the identity provider.

If there is no WAYF cookie present, the service provider must invoke the active WAYF process. Just as SSO profiles allow for push and pull variants, so does WAYF processing. The WAYF pull variant has a service provider presenting the user with a list of (trusted) identity providers for the user to select from. The WAYF push variant has the service provider presenting the user with a notice to attempt to SSO from their IdP (using a push-based SSO). The WAYF push variant may be employed in situations where a service provider is not able to advertise all of their trusted identity providers (for competitive reasons, for example).

3.3.4 Integrating SSO with Access Manager for e-business

Tivoli Federated Identity Manager provides the run-time implementation of supported SSO profiles. Access Manager for e-business provides the HTTP point of contact functionality. As such, Tivoli Federated Identity Manager has dependencies on Access Manager for e-business, and Access Manager for e-business has dependencies on Tivoli Federated Identity Manager. In this section, we briefly discuss these interdependencies.

Tivoli Federated Identity Manager relies on the point of contact for session management for all users, whether Tivoli Federated Identity Manager is acting at the identity provider or service provider. As part of the user's session management, Access Manager for e-business will be responsible for only allowing authorized users to participate in SSO relationships (for example, not all of an identity provider's users may be entitled to F-SSO functionality).

When configured in an identity provider environment, Tivoli Federated Identity Manager expects that Access Manager for e-business will correctly authenticate users, and will assert the user's identity to Tivoli Federated Identity Manager as part of a SSO request. This implies that from an Access Manager for e-business point of view, access to the Tivoli Federated Identity Manager SSO endpoints must be treated as protected resources.

When configured in a service provider environment, Tivoli Federated Identity Manager must be able to determine a user's local identity and create an Access Manager for e-business credential for this user.

3.3.5 F-SSO approaches

F-SSO may use a variety of methods to communicate and assert identity. The different methods will not have support for all functionality described in 3.3.3, "F-SSO protocol functionality" on page 105. The standards were introduced in 2.3, "FIM standards and efforts" on page 51, and some of the characteristics of each protocol are highlighted in Table 2-1 on page 58. For detailed examples this book describes use cases for each of these SSO solutions in Part 2, "Customer environment" on page 181. In general, aside from proprietary solutions, there are three approaches to Web-based browser single sign-on and federation:

- ▶ SAML
- ▶ Liberty ID-FF
- ▶ WS-Federation

SAML

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: The SAML assertion (used to transfer information about a user) and the SAML protocol (the means of exchanging a SAML assertion). Full details on SAML are available from:

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

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 (voucher or token) is exchanged between an identity provider and service provider.

With a Browser/POST profile, a SAML assertion (voucher or token) is included in the response that is sent to the service provider as part of an HTML form as in Figure 3-13. This is a *front channel* exchange of the SAML assertion.

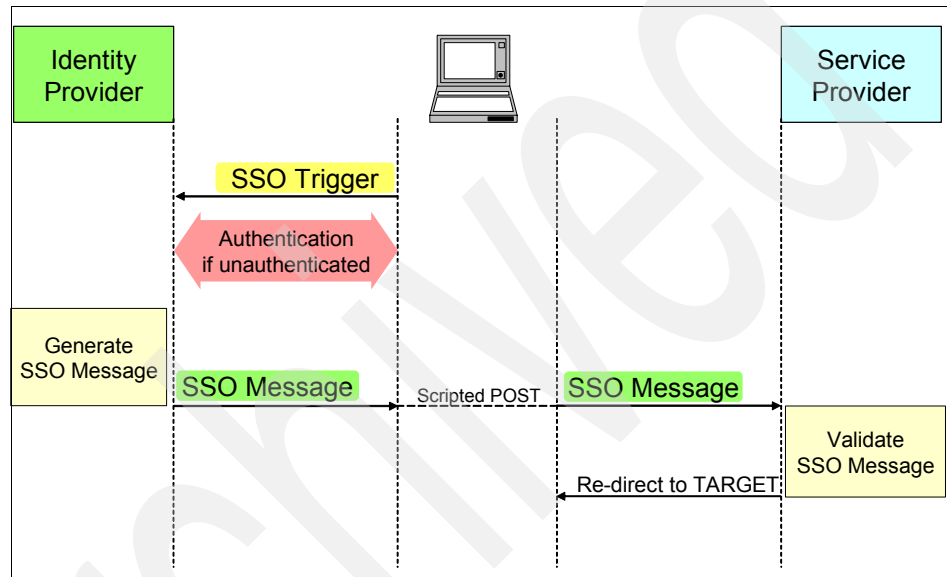


Figure 3-13 SAML SSO: Browser POST

With a Browser/Artifact profile, a pointer to the SAML assertion (called an *artifact*) is included in the query_string of an HTTP 302 redirect 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 will be a URL that corresponds to a FIM endpoint.

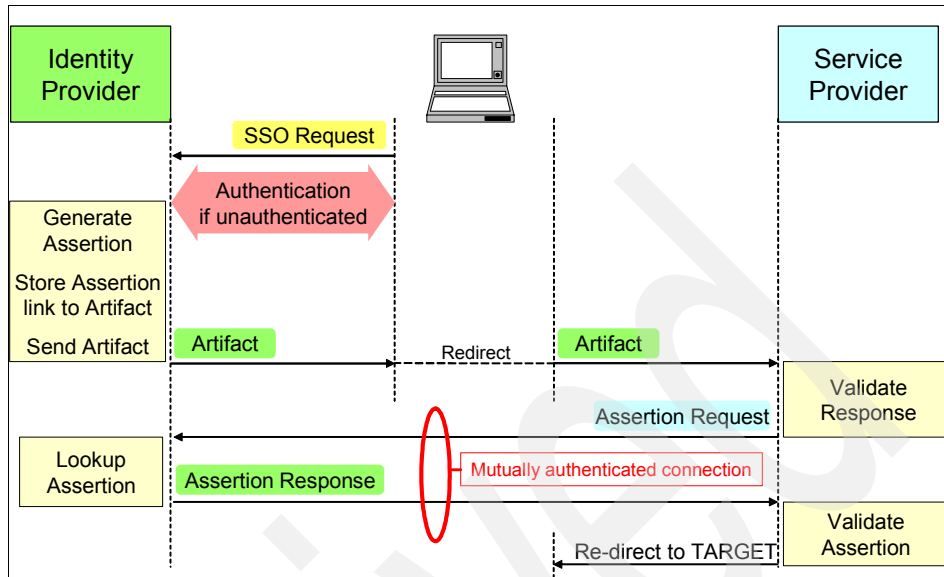


Figure 3-14 SAML SSO: Browser/Artifact

In Figure 3-14, 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.

Liberty ID-FF

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 single sign-on protocols
- ▶ Functionality for single logout (SLO)
- ▶ Account linking and de-linking:
 - Liberty Register Name Identifier profile (RNI)
 - Liberty Federation Termination Notification profile (FTN)
- ▶ *Where are you from?* (WAYF)
 - Liberty identity provider introduction profile (IPI)
- ▶ Unsolicited authentication response

- This 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)

Details of the Liberty profiles are given in the following Liberty Alliance specifications: [*liberty-architecture-bindings-profiles-v1.1*] and [*liberty-architecture-protocols-schema-v1.1*], and:

<http://www.projectliberty.org/>

Browser/Artifact single sign-on profile

The flows of the Liberty Browser/POST single sign-on profile are shown in Figure 3-15.

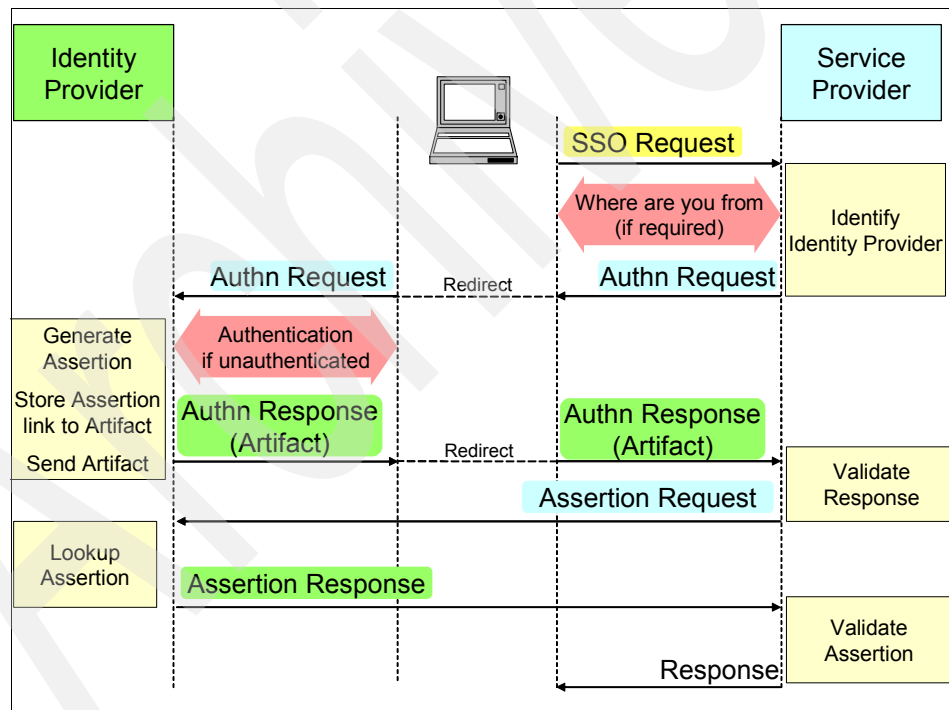


Figure 3-15 Liberty: Browser POST profile

In this profile the identity provider sends the Liberty Assertion (or SAML status message) in the Authentication Response.

Note the Where Are You From (WAYF) functionality embedded in this SSO profile. This is required so that the service provider can figure out which identity provider it should direct the client to in order to obtain a Liberty Assertion. This might involve reading information from a previously stored cookie, or it might require interaction with the user to prompt for the appropriate identity provider.

In order to generate a Liberty Assertion for the client, the identity provider must have an authenticated session. If the session is not already authenticated when the Auth Request arrives then the identity provider needs to authenticate the user at that point. Note that some options in the Auth Request may prevent the identity provider from authenticating the user. If this is the case then the identity provider will send an error in the Auth Response.

The Auth Response in this profile is sent in an HTML form. Scripting is included so that the form is automatically POSTed to the service provider.

Liberty Register Name Identifier (RNI)

The Liberty Register Name Identifier 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 (FTN)

The Liberty Federation Termination Notification 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 Sign-Out (SLO)

The Liberty Single Sign-Out profile defines the process by which a (set of) valid session(s) for a user is destroyed. Single sign-out 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 identity provider, where the IdP in turn logs the user off of all currently valid SP sessions *except* the SP session that initiated the IdP logout.

Note that while sign-out is almost always an end-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 end user. This may 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 may be leveraged, as it may occur out-of-band (without waiting for a user interaction at either side).

Identity provider introduction (IPI)

The Liberty identity provider introduction 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 accessible/required by all business partners within the circle of trust, in particular, the user's identity provider. Once retrieved, the information contained in the cookie is extracted and returned to the requested domain using techniques such as URL re-writing.

Liberty-enabled client/proxy (LECP)

The Liberty-enabled client/proxy profile is designed to address devices that are not able to accommodate the query-string length requirements of the B/A profile or the form post requirements of the B/P profile. These devices are generally mobile devices, such as query-string length limited mobile devices or older mobile devices 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 wishes to use with the service provider. This 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 3-16 shows the role of Tivoli Federated Identity Manager in a LECP profile, where a WAP Gateway is acting as the LECP. Note that in this scenario, Tivoli Federated Identity Manager need only accommodate steps 4 and 6 when acting as an identity provider, and steps 1, 3, 7, and 11 when acting as a service provider.

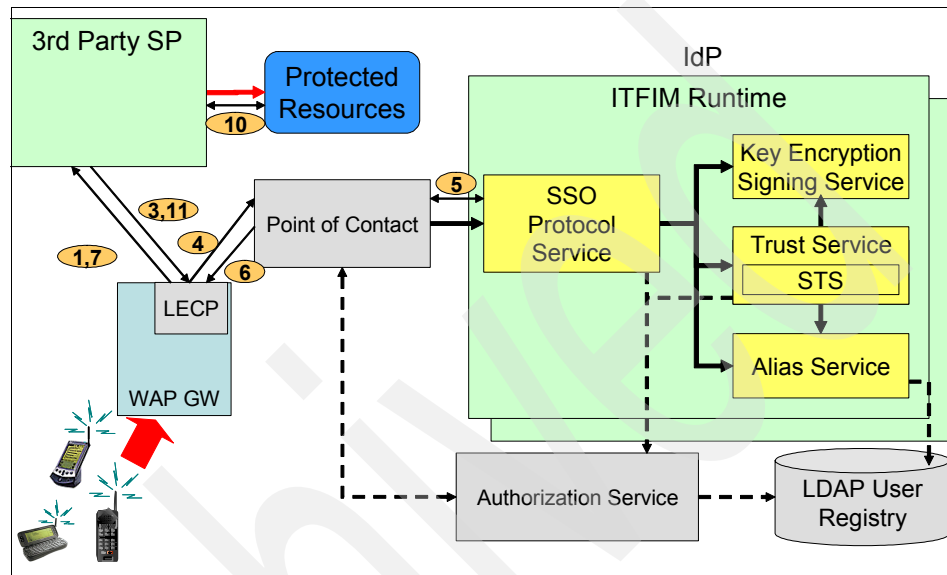


Figure 3-16 Liberty enabled client proxy (LECP) example

Details of the LECP profile are given in the following Liberty Alliance specifications: [[liberty-architecture-bindings-profiles-v1.1](#)] and [[liberty-architecture-protocols-schema-v1.1](#)].

Unsolicited authentication response

This is how Liberty ID-FF 1.2 does a *PUSH* SSO, and, yes, we should have a section on this.

Liberty 1.2 allows for an identity provider to send an *unsolicited authentication response* to a service provider. This allows a push SSO to take place—an SSO initiated by the identity provider. The trigger for this is not specified so it is up to who implements it to decide.

WS-Federation passive client

The WS-Federation passive client specification, published by IBM and Microsoft, available at <http://www-106.ibm.com/developerworks/library/ws-fedpass/>, states that:

The WS-Federation specification defines an integrated model for federating identity, authentication, and authorization across different trust realms and protocols. This specification defines how the WS-Federation model is applied to passive requestors such as Web browsers that support the HTTP protocol.

The WS-Federation allows for both pull and push for SSO.

- ▶ Pull means that the SSO is initiated at the service provider, the service provider determines the identity provider, then the service provider requests SSO from the identity provider and the identity provider responds with an SSO token. See Figure 3-17 on page 118.
- ▶ Push means that the SSO is initiated at the identity provider and then the identity provider sends the SSO token to the service provider. See Figure 3-18 on page 119.

Pull

In Figure 3-17 on page 118 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 then 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.

Once 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 re-redirect 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 will allow 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 redirect from the service provider.

When the identity provider receives the SSO Request at its WS-Federation endpoint, it will first authenticate the user (if they are currently unauthenticated). It must have an authenticated session in order to process a single sign-on request. The identity provider reads the SSO request from the service provider and builds an appropriate SSO response message for that provider. This message will include 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.

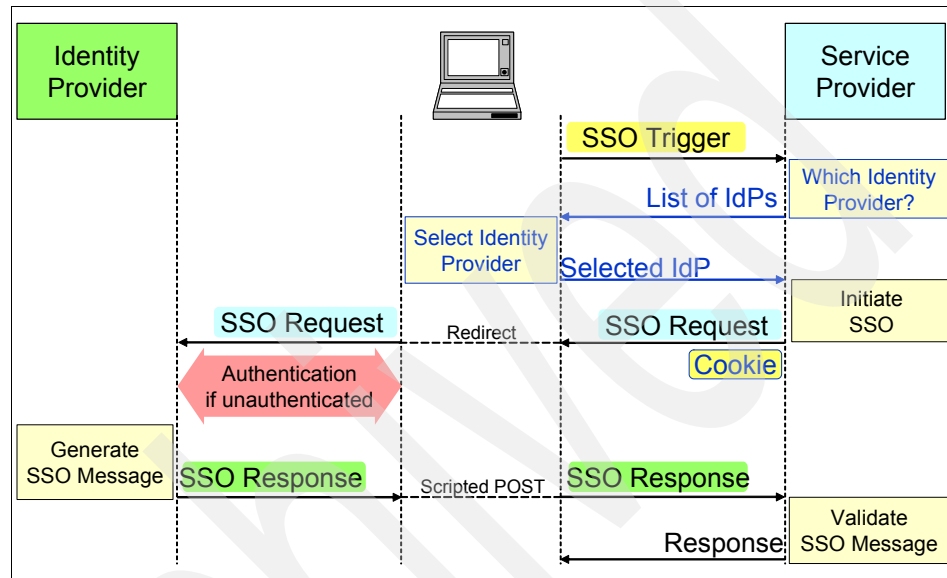


Figure 3-17 WS-Federation: Select ID Provider and SSO (Pull)

Push

Figure 3-18 on page 119 shows the protocol flow for a WS-Federation PULL 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.

It is unlikely that a user would manually type an SSO Request message into their browser (although they could); it is much more likely that an identity provider will include a link on their site that a user can select in order to access some service provider resource (for example, For BigCorp you would see the message `Click here to book a hotel with our preferred partner RBTravel`). 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 had it 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 to the service provider, via the client, using a HTML Form.

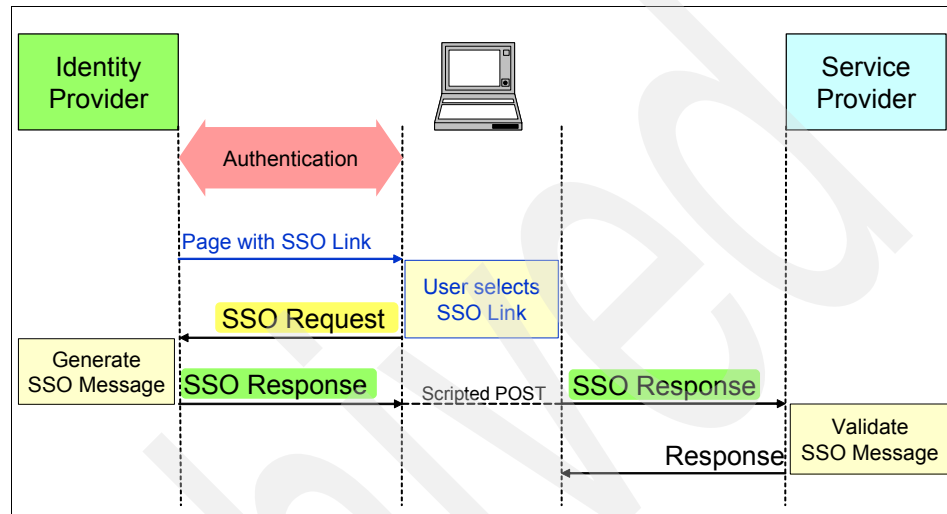


Figure 3-18 WS-Federation: SSO (Push)

WS-Federation also supports Single Sign-out at both the SP and IdP.

3.3.6 InfoService

The InfoService is used to build a user interface reflecting the users' defined federations. If a portal has many services where users have the possibility to use F-SSO then it is necessary to be able to present the choices in a relevant manor, as not to confuse the users.

The Info Service provides an interface that can be used to determine a user's federations. This then allows customized and personalized Web pages, listing the sites to which the user can SSO, and presenting the list of sites to which the user can federate (and subsequently SSO). This can also be used to control the presented interactions, such as when de-federation is presented as a possible action (so that a user is not given the option of de-federating from a provider to whom they have not federated in the first place).

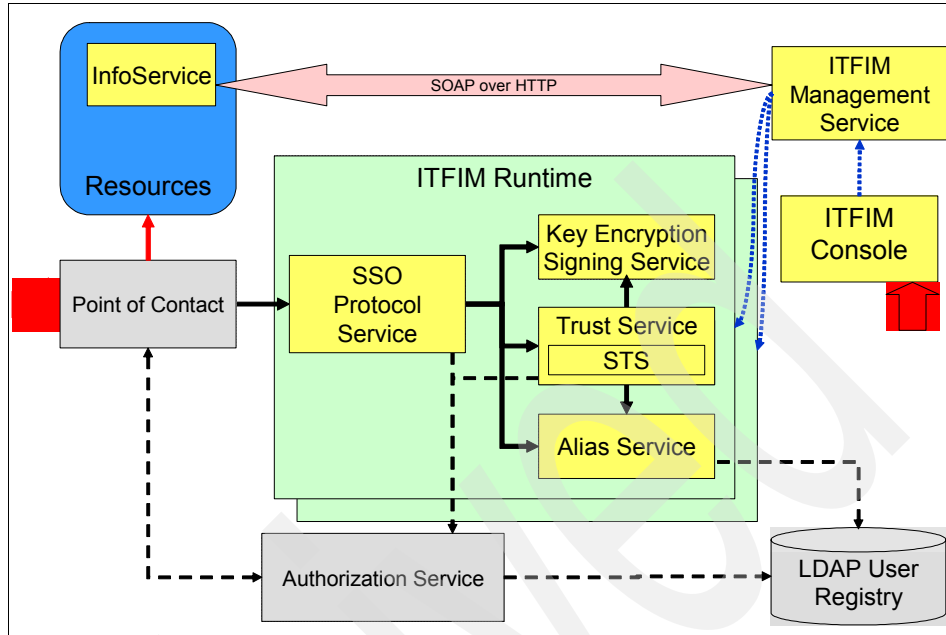


Figure 3-19 Tivoli Federated Identity Manager InfoService access to the Management Service

The InfoService makes Web services calls to the Management service to get this information. See Figure 3-19.

For an example of how the InfoService is used see Chapter 3, “Tivoli Federated Identity Manager architecture” on page 85.

3.3.7 Specified level view of F-SSO architecture

There are many ways to deploy a F-SSO solution. This pattern gives an attempt to show how it could be accomplished.

The specified view for a IBM Tivoli FIM architecture for F-SSO is shown in Figure 3-20 on page 121. A specified view describes the key nodes and the connections between them.

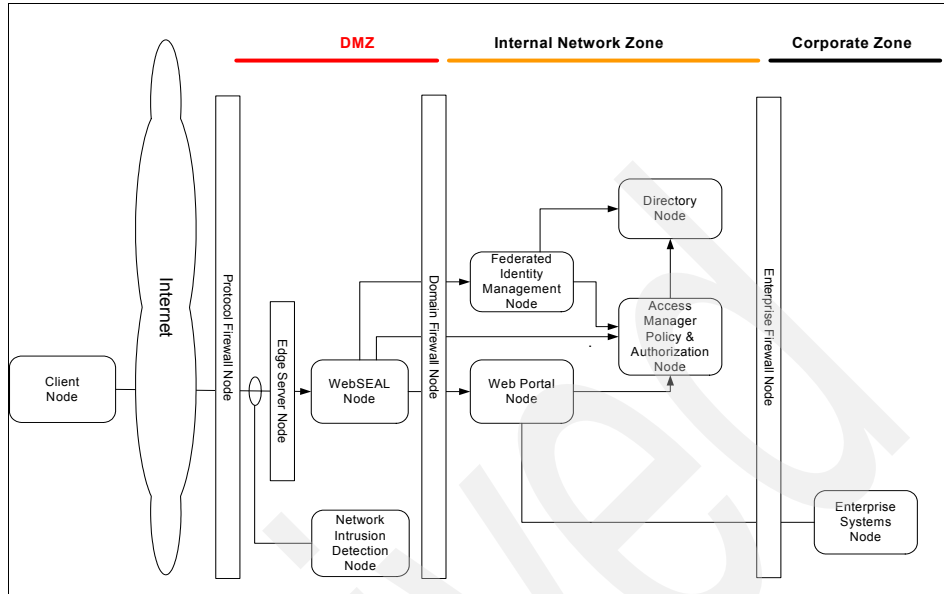


Figure 3-20 Generic IBM Tivoli FIM specified level view of F-SSO

A more detailed look at F-SSO deployment is available in 4.1, “Federated SSO architecture patterns” on page 136.

3.4 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.

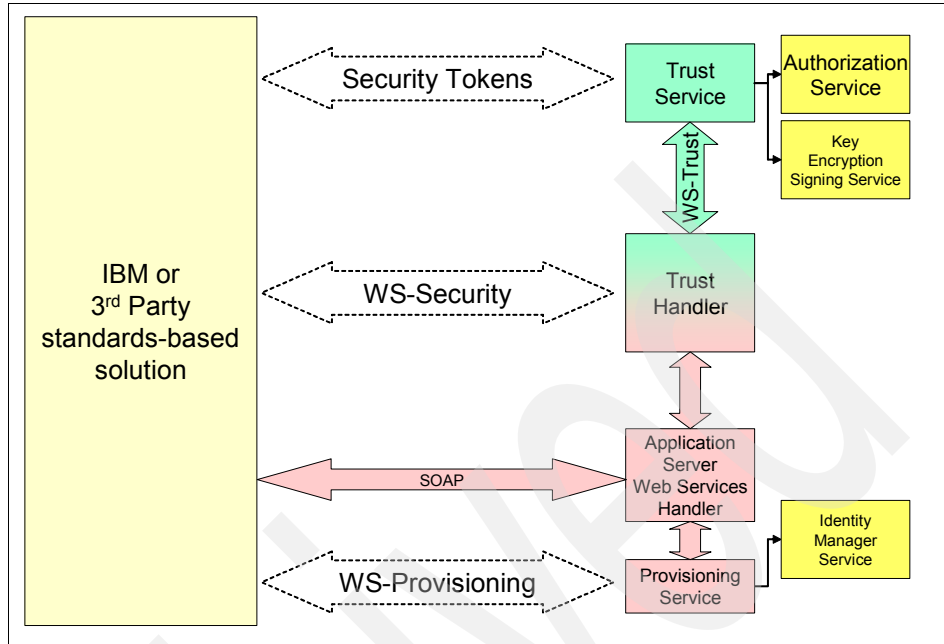


Figure 3-21 Web services security: Components and communication

Figure 3-21 shows the communications and exchanges that take place at each layer of Tivoli Federated Identity Manager when performing Web services security management.

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 via 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).

3.4.1 Architecture overview

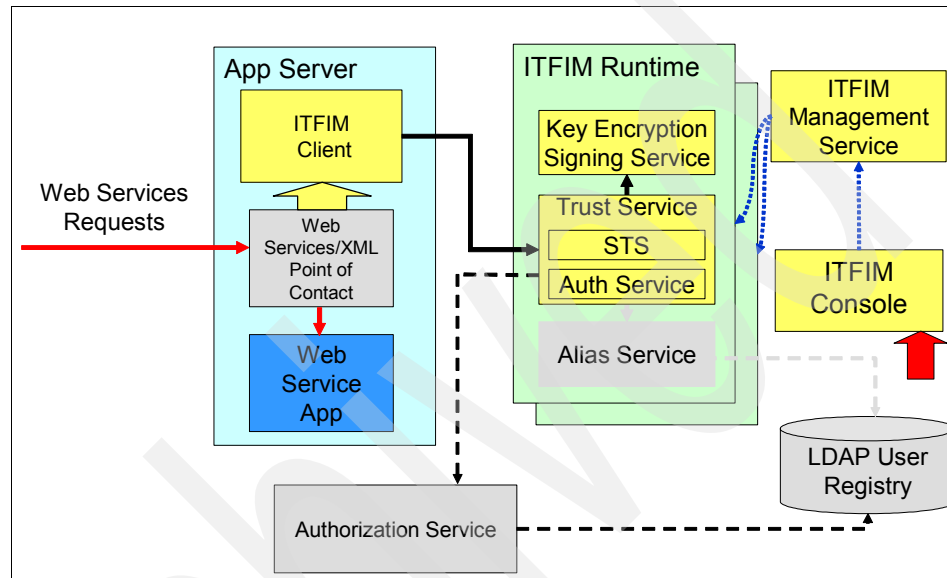


Figure 3-22 Components for Web services security management

Figure 3-22 shows the components required for Web services security management with Tivoli Federated Identity Manager.

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.

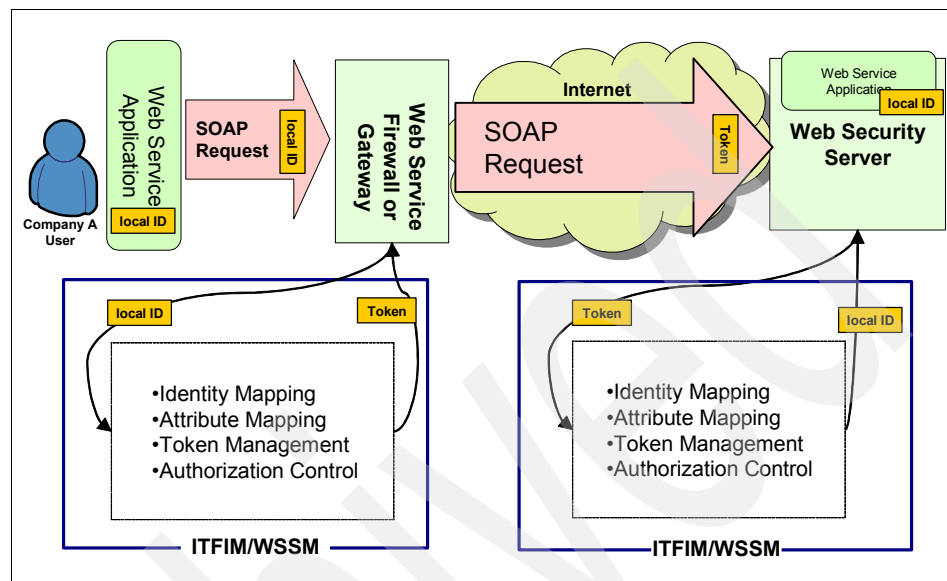


Figure 3-23 Web service security management (WSSM): Solution architecture

Figure 3-23 shows a user at company A accessing a resource at company B via a Web service request.

1. User at company A invokes a Web service using her 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 such that they can be processed by the receiving company B. Its functionality may include:
 - Mapping of identity claimed in incoming locally valid ID to a token
 - Mapping of local valid attributes such as groups/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 a number of different 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 will do 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 TFIM authorization service.

3.4.2 WS-Security

WS-Security is used to accomplish end-to-end message security. Message-based security 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

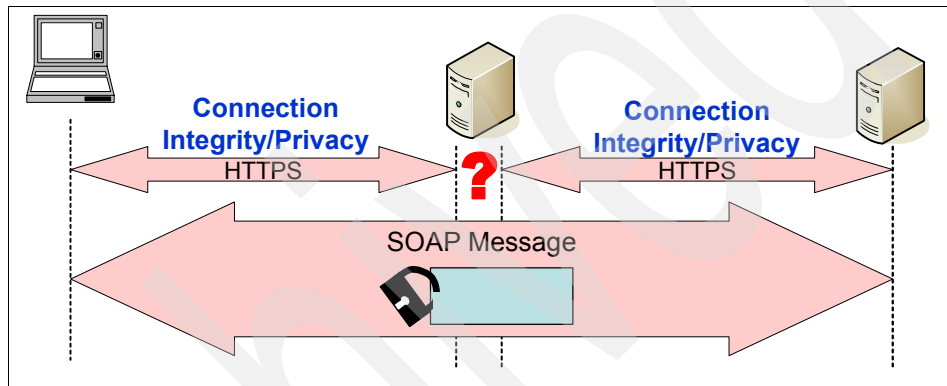


Figure 3-24 Message-based security: End-to-end security

In Figure 3-24 end-to-end message security is illustrated. 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 2.5.2, “Web services security” on page 68.

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 in and of itself is tamper-proof and confidential.

3.4.3 Web services Gateway or Firewall

A Web services gateway or firewall is much the same as a HTTP Reverse proxy. A WS Gateway enables the company to separate internal network topology from the Internet, allowing for flexibility and abstraction.

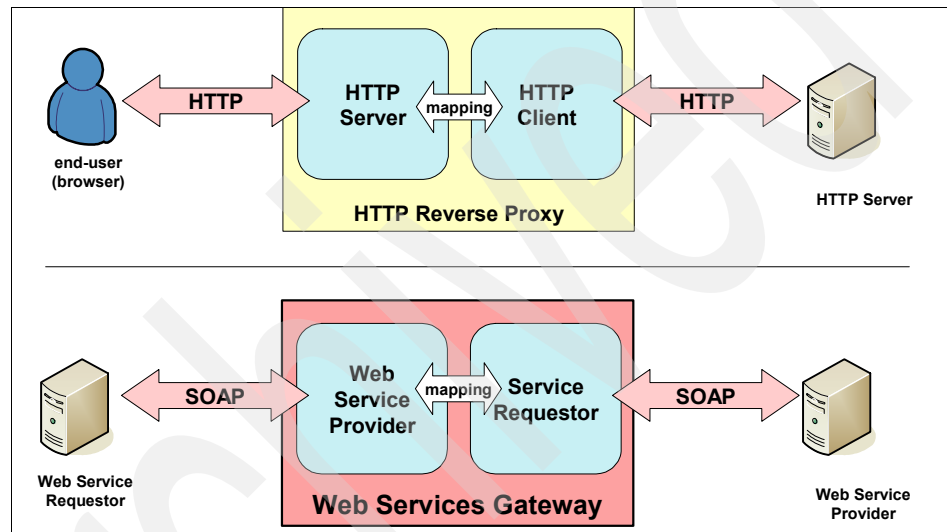


Figure 3-25 Web services gateway: A reverse-proxy for Web services

Challenges that are addressed by the Web services Gateway are:

- ▶ Decouple deployment from invocation

Separate the actual implementation of a service from how another service accesses it. These include:

- 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 need the flexibility of changing your deployment infrastructure without notifying all the service requestors. Say a Web service is deployed in a machine that later fails during operation. There

needs to be a process 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 may be using different protocols to invoke your Web service. You need a mechanism to reconcile the different service invocations to match the needs of the internal infrastructure.

For more details on 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 is a way point for 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 on Web services gateway see Chapter 4, “Deploying Tivoli Federated Identity Manager” on page 135.

3.4.4 WS-Trust

The WS-Trust specification defines the interface used to manage the security tokens defined by the WS-Security specification. The TFIM 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 conformant 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 on the trust service see 3.2.3, “Trust services” on page 92.

3.4.5 Authorization services (AS)

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 if a user (as validated and identified by the trust service) is authorized to access requested resources. This 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, a CICS resource, or some other proprietary resource.

3.4.6 Web services security management architecture approach

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

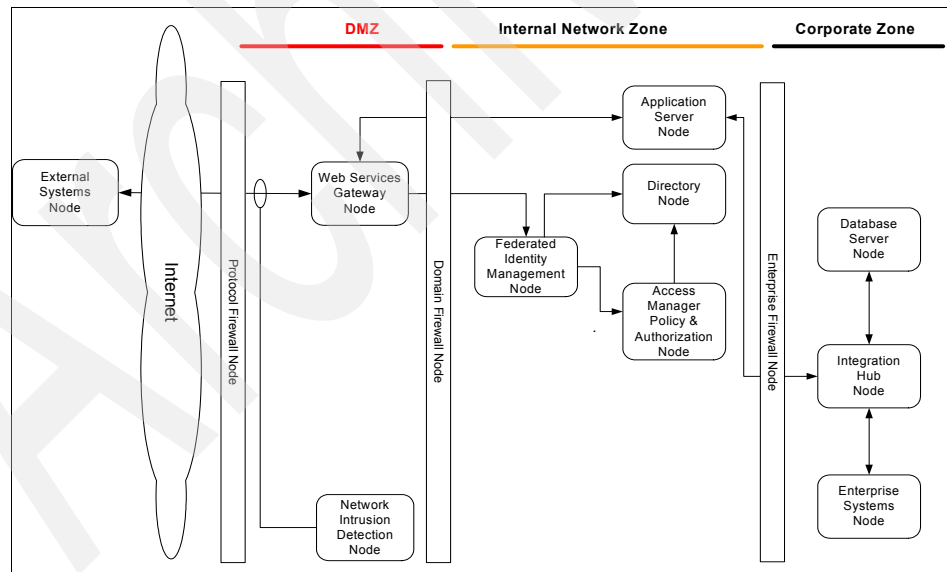


Figure 3-26 Specified level view of Web services security management

A more detailed look at Web services security management deployment is available in In 4.2, “Federated Web services architecture patterns” on page 151.

3.5 Provisioning services

Provisioning services are used within a federated environment for both a priori and run-time provisioning solutions, as described in 2.6, “Federated identity provisioning” on page 70. Provisioning services interact with both local identity management systems (such as Tivoli Identity Manager) and local data stores (access via 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; see 2.2.5, “Identity attributes” on page 45.

There are few widely accepted standards for provisioning. The most important effort to date is probably the work done by the Provisioning Service Technical Committee (PSTC) at OASIS. The PSTC has defined a set of use cases that reflect the operational requirements of a provisioning system. WS-Provisioning is compatible with those use cases.

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.

The specification defines a model for the primary entities and operations common to provisioning systems including the provisioning and de-provisioning of resources, retrieval of target data and target schema information, and provides a mechanism to describe and control the life cycle of provisioned state.

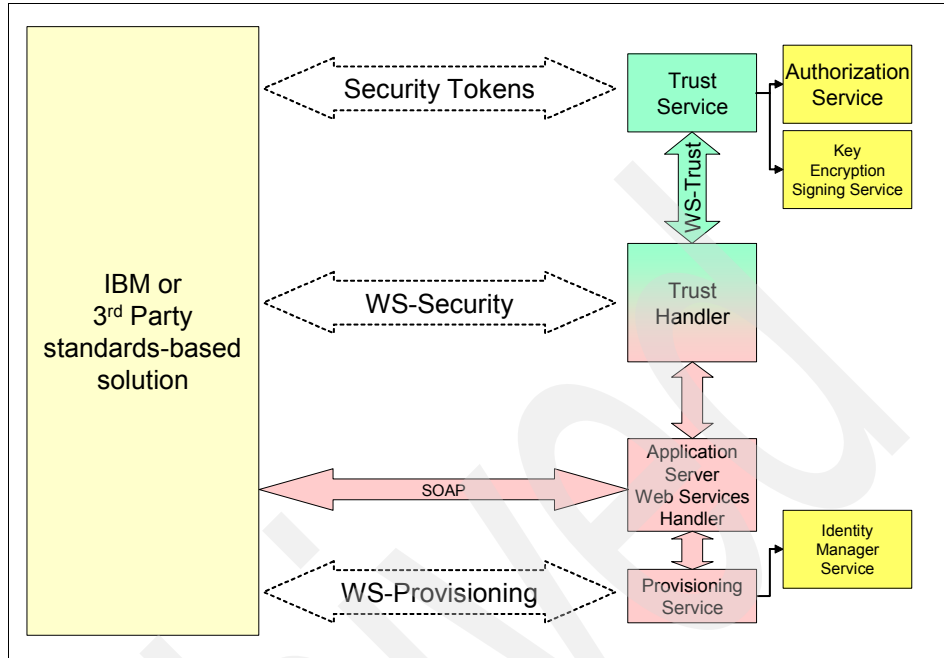


Figure 3-27 Web services provisioning: Components and communication

Figure 3-27 shows the communications and exchanges that take place at each layer of Tivoli Federated Identity Manager when performing Web services provisioning.

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 via 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.

3.5.1 Architecture overview

Figure 3-28 shows the components required in order to implement secure, cross-enterprise provisioning using Tivoli Federated Identity Manager.

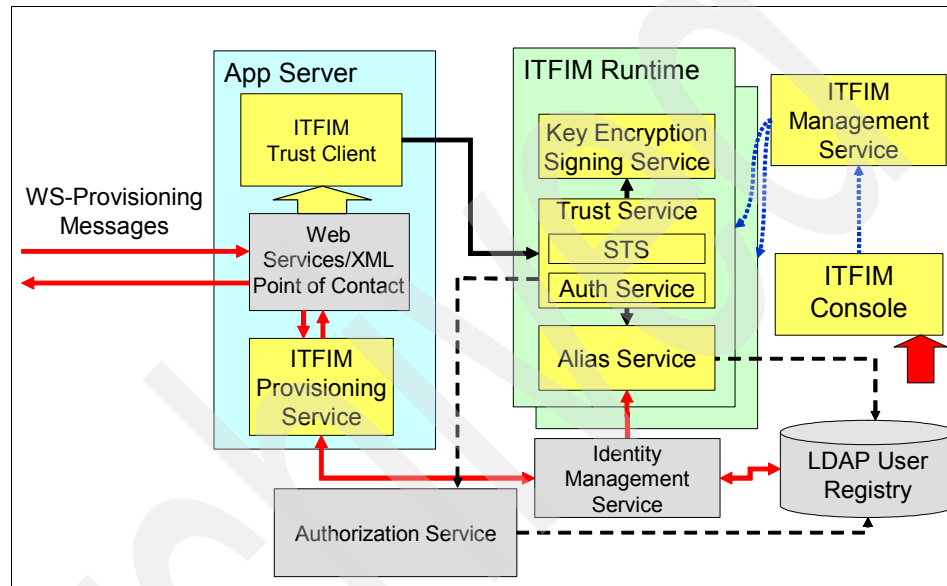


Figure 3-28 Components for federated user provisioning

It is important to note here that many of the components shown here are the same as required to secure any Web service. The provisioning service is just another Web service in that respect.

The only components specifically related to provisioning are the provisioning service itself and Identity Management Service, which is the enterprise Identity Management Service, in this case the IBM Tivoli Directory Integrator (TDI), 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.

WS-Provisioning messages are received by the application server Web services handler, in this case the WebSphere Services handler, and are authorized using Tivoli Federated Identity Manager and authorization service, here Tivoli Access Manager. If authorized, the request is passed on to the Tivoli Federated Identity

Manager provisioning service. The provisioning service validates the request and then passes it on to ITDI. An ITDI assembly line extracts the identity information from the provisioning request and handles as appropriate. If the request is to provision a local account for Liberty SSO then the Alias Service is called to associate the newly created user with the received Liberty alias.

Although the diagram above shows ITDI interfacing directly to the LDAP user registry, this is just an example. ITDI could be configured to interface with any supported endpoint including IBM Tivoli Identity Manager.

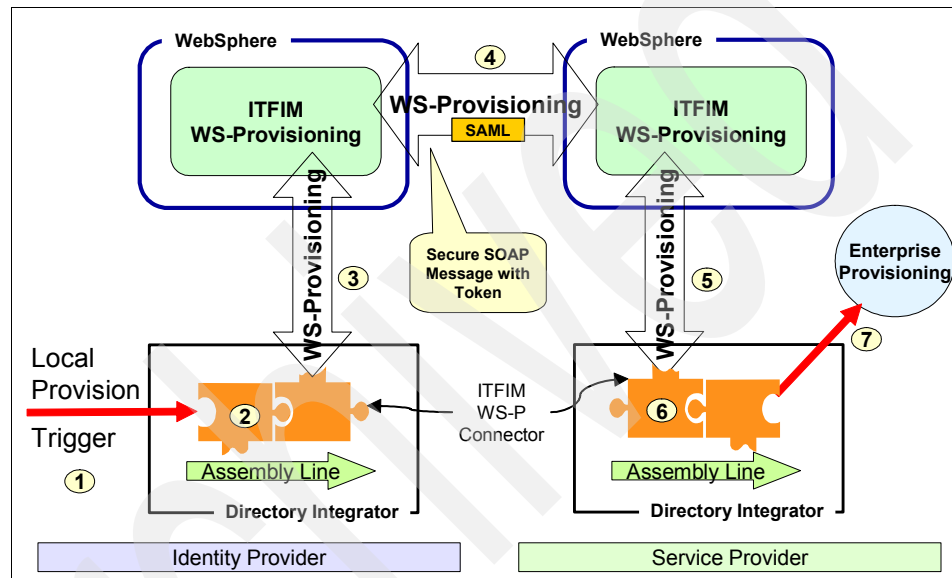


Figure 3-29 Federated provisioning - Overview

Figure 3-29 provides an overview of the WS-Provisioning support provided in Federated Identity Manager. The Tivoli Federated Identity Manager components are:

- ▶ The Tivoli Federated Identity Manager WS-Provisioning Web service that runs on WebSphere Application Server 6.0
- ▶ The Tivoli Federated Identity Manager WS-Provisioning Connector that runs on IBM Tivoli Directory Integrator

Both of these 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 via this sequence:

1. Some type of *provisioning trigger* at the IP initiates a Tivoli Directory Integrator assembly line. Tivoli Directory Integrator provides several mechanisms to start an assembly line. 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 assembly line collects data to form a WS-Provisioning message. The assembly line 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 assembly line.
7. The Tivoli Directory Integrator assembly line 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 that the 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.

3.5.2 Provisioning architecture approach

There are many ways to deploy a provisioning solution. This view gives an attempt to show how it could be accomplished leveraging Web services security management.

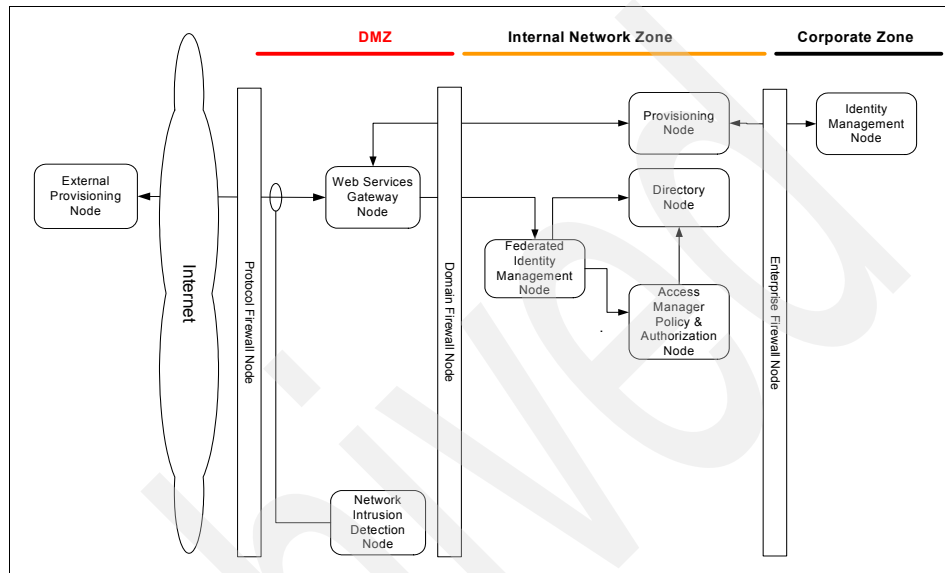


Figure 3-30 Generic IBM Tivoli FIM specified level view of provisioning

3.6 Conclusion

At the beginning of this chapter we discussed the federated identity management functionality and how that functionality consists of a set of services. Then we described three solution area (F-SSO, Web services security management, and provisioning) studying functional details within each solution area.

The focus of the chapter was to give a description of how the Tivoli Federated Identity Manager solution is implemented to meet the overall FIM challenge. We discussed how the Tivoli Federated Identity Manager solution is built around the trust infrastructure implemented by the trust service. Single sign-on services provide the implementation of federation protocols, and also the interface between the point of contact (PoC) and the trust service.



Deploying Tivoli Federated Identity Manager

This chapter describes architecture options for deploying Tivoli Federated Identity Manager, approaches for integrating Tivoli Federated Identity Manager with other middleware and customer applications, and several important issues relating to deploying Tivoli Federated Identity Manager in a production environment.

4.1 Federated SSO architecture patterns

Tivoli Federated Identity Manager is a flexible product that provides a federated identity management solution for both browser-based single sign-on and Web services environments. As there are many different examples of environments that require a federation solution, there are many different ways that Tivoli Federated Identity Manager can be deployed. We can represent the deployment of Tivoli Federated Identity Manager with several typical deployment/architecture patterns. In this section, we describe the most patterns from which customer-specific deployments can be generated.

4.1.1 Architecture approach

Tivoli Federated Identity Manager's federated single sign-on (F-SSO) solution enables the single sign-on of a user in a cross-Enterprise, or cross-domain, scenario. Tivoli Federated Identity Manager's F-SSO functionality does *not* replace an Enterprise's existing authentication and session management services, nor any of the sign-on functionality they provide to the Enterprise's applications. Tivoli Federated Identity Manager's F-SSO solution handles SSO to an edge-based *point of contact* component. This is based on the underlying principal that because Tivoli Federated Identity Manager does not replace existing session management functionality, it should not directly provide single sign-on to individual applications within an Enterprise (Enterprise single sign-on).

An architectural model based on a (scalable, available, performant) point of contact provides many security benefits, including the ability to control all access to an environment, closing off "back doors" that allow unauthorized users to access an Enterprise's environment. Typically, an edge component, such as Access Manager for e-business, acts as a point of contact and is used to provide single sign-on from Tivoli Access Manager for e-business (where the user's authentication credentials are collected and evaluated) to individual backend applications. This functionality remains unchanged by the addition of a Tivoli Federated Identity Manager solution.

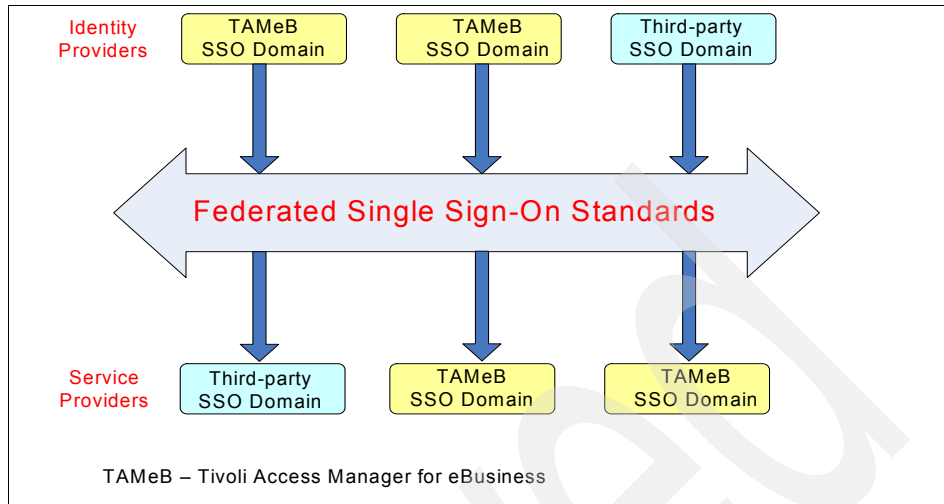


Figure 4-1 Linking SSO domains with Federated SSO protocols

This architectural approach to Federated SSO provides the following advantages:

- ▶ Little or no changes are required to Enterprise applications.
- ▶ Lightweight SSO within a domain.
- ▶ Support for identity provider applications.
- ▶ Able to leverage existing Tivoli Access Manager for e-business infrastructure.

Little/no change to applications

Many toolkit-based offerings for Federated SSO require fairly intrusive modifications to the applications to call the (proprietary) product APIs required to implement Federated SSO. These toolkit approaches are typically marketed as lightweight approaches; however, in terms of total project costs and maintenance costs, they are often more expensive than middleware solutions (such as Tivoli Federated Identity Manager), even for small-to-medium size deployments. These so-called lightweight solutions can be even more expensive if an environment does not have existing session management functionality; many federation solutions assume that this type of functionality exists and can be leveraged as part of an F-SSO solution for single sign-on and single (federated) logoff.

The Tivoli Federated Identity Manager approach leverages Access Manager for e-business's ability to provide SSO to an application with few or no changes to the application. For those applications that use underlying middleware functionality to manage authentication, the middleware container can usually be configured to accept the user identity from Access Manager for e-business without any changes to the applications using the authentication data. For

example, the IBM WebSphere Application Server provides a feature called a Trust Association Interceptor (TAI) to accept a user ID from an HTTP header variable and create a login context for that user. Most other middleware products have similar functionality. For those applications that implement their own custom authentication logic, a small change to the login module to accept the user identity from a HTTP header variable, rather than prompting the user for a user ID and password, is typically fairly straightforward to code and test.

The Tivoli Federated Identity Manager approach provides a loose coupling between the application and the Federated SSO functionality and avoids the use of proprietary APIs.

Lightweight SSO within a domain

The digital signing and validation of XML-based assertions, such as those used in the Federated SSO protocols, involve encryption and decryption using relatively long asymmetric keys. Such operations incur a fair degree of computational overhead. This computational overhead is required (and thus accepted) as part of the proof of a trust relationship governing federated single sign-on. The trust relationship between a *point of contact* (for example, Access Manager for e-business) and back-end protected applications does not normally require techniques that are as costly. For example, these internal trust relationships can be based on techniques such as mutually authenticated SSL or known internal IP addresses.

By using a lightweight SSO technique between Access Manager for e-business and the (possibly hundreds of) protected applications within an Enterprise, this overhead is only incurred where it is needed—in those cases where we need to provide SSO from one domain/organization to another. The Tivoli Federated Identity Manager approach therefore provides a more efficient and scalable architecture and a more responsive user experience when working with multiple applications within a domain.

Support for identity provider applications

Even for pure identity provider deployments (no local services/protected resources are made available to the user), there are often self-care and portal applications associated with the identity provider's identity management functionality. The use of Access Manager for e-business to provide the *point of contact* for the identity provider leverages the (lightweight) SSO facilities of Access Manager for e-business to access the identity provider applications without incurring the overhead of running and accessing a separate service provider site for those applications.

Leverage existing Access Manager infrastructure

For those customers who have already deployed a Access Manager for e-business SSO infrastructure, upgrading it to provide Federated SSO functionality is a relatively straightforward exercise. Moreover, in most cases the applications will not require any modification, thereby significantly reducing the time and costs needed to deploy the Federated SSO functionality.

4.1.2 Base pattern

The Base architecture pattern for deploying Tivoli Federated Identity Manager for Federated SSO uses the reverse proxy component of Access Manager for e-business (WebSEAL) to provide the *point of contact* for Tivoli Federated Identity Manager, namely authentication (at the identity provider side) and session management (for both an identity provider and service provider deployment). In this Base pattern, all users who use the Federated SSO functionality are individually defined in the Access Manager for e-business user registry².

On the identity provider side of a federation, Access Manager for e-business (WebSEAL) manages the local user authentication process, using any of its supported authentication mechanisms. WebSEAL manages the user's session, including (optionally) brokering access to the identity provider's protected applications based on Access Manager for e-business managed access control policies. Note that these policies can be as simple as access is allowed based on successful authentication, to more complex, such as access is allowed (or disallowed) based on a user's group membership, roles, or other attributes (entitlements).

If a user requests single-sign-on (or has it requested on their behalf by a service provider partner), Access Manager for e-business will pass control to the Tivoli Federated Identity Manager server. Note that Tivoli Federated Identity Manager itself, and the single sign-on functionality, can be access controlled by Access Manager for e-business. This has the effect of allowing a customer (in a more advanced deployment) to provider single sign-on functionality to a subset of its users. Included with this request to Tivoli Federated Identity Manager will be the user's local (Access Manager for e-business based) identity. The Tivoli Federated Identity Manager server will use this identity for the building of the assertion provided as part of a single sign-on response.

² While this discussion focuses on the use of the Access Manager for e-business reverse proxy (WebSEAL), it is equally possible to provide point of contact functionality using the Access Manager for e-business Web server plug-in. The plug-in approach is described in the next section.

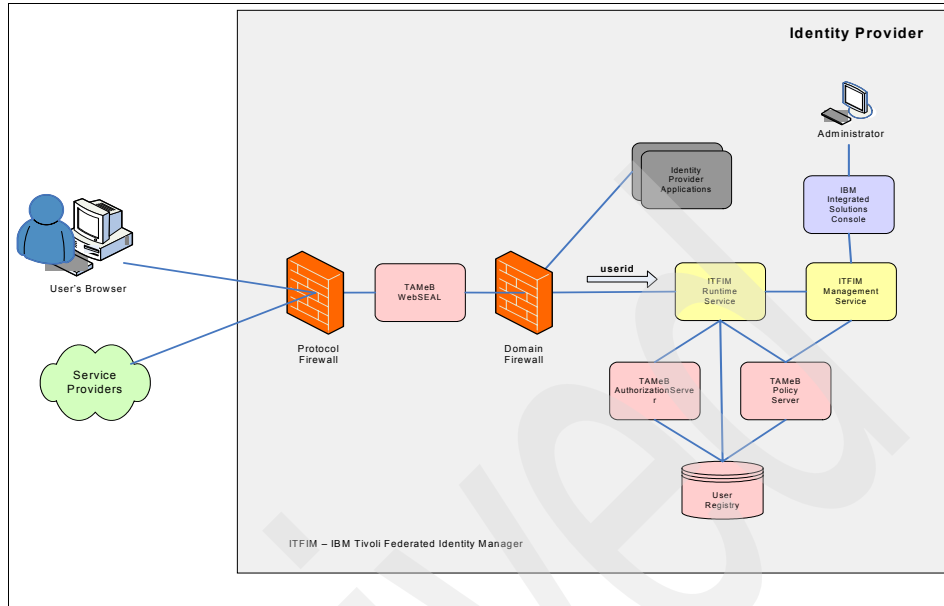


Figure 4-2 Base pattern for identity provider

For a service provider configuration, Access Manager for e-business (WebSEAL) is configured to allow unauthenticated access to the Tivoli Federated Identity Manager application, namely the login endpoint associated with the federation. Once Tivoli Federated Identity Manager has successfully validated and processed the incoming SSO message, it creates an Access Manager for e-business credential and passes it back to the WebSEAL server via the Access Manager for e-business External Authentication Interface (EAI). This allows WebSEAL to establish and manage an authenticated session for the user. See Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, for a description of the External Authentication Interface of Access Manager for e-business.

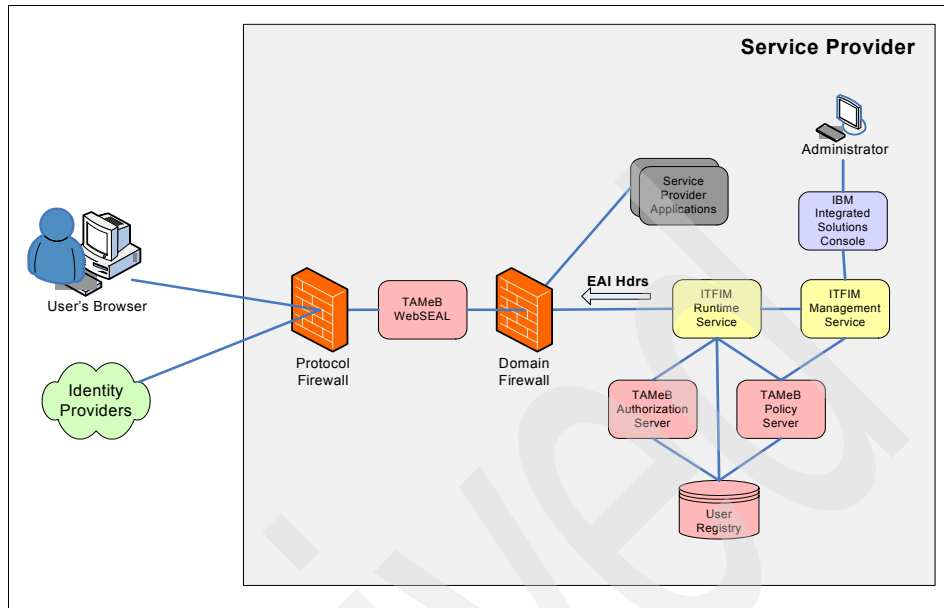


Figure 4-3 Base pattern for service provider

Several Federated SSO protocols include SOAP-based profiles. These profiles are used to retrieve information from a back-channel (directly between the identity provider and the service provider, without redirection via the user's browser). This back-channel communication is (confidentiality) protected through the use of SSL. The SOAP traffic is sent over SSL and Tivoli Federated Identity Manager will validate the (SSL) X.509 server certificate presented by the server hosting the SOAP endpoint.

The use of SSL does not provide authentication of the requestor (initiating the SOAP request). Additional techniques are required for authentication purposes:

- ▶ Rely on message level authentication.
- ▶ Rely on channel level authentication.

As message level authentication provides no additional burden on the Tivoli Federated Identity Manager servers, the Tivoli Federated Identity Manager SOAP endpoint is configured to use the same set of replicated WebSEAL servers as the login endpoint.

When additional channel-level authentication is required, mutually authenticated SSL techniques are required. The service provider presents an X.509 client certificate to the identity provider during the establishment of the SOAP connection. This allows a mutually authenticated SSL session to provide both

authentication of the service provider and protection of communications in transit.

When a mutually authenticated SSL type solution is required, a dedicated set of replicated WebSEAL servers is required at the identity provider. These WebSEAL servers listen on a different IP address and/or different port than the main set of WebSEAL servers yet are junctioned to the same Tivoli Federated Identity Manager servers as the main set of WebSEAL servers. These additional servers are configured to request and validate an X.509 client certificate as part of the HTTPS session establishment. These extra WebSEALs are then governed by a different trust relationship from the typical HTML/HTTP serving WebSEALs. In particular, these SOAP-accessible WebSEALs can provide a stronger trust relationship between the identity provider and service provider.

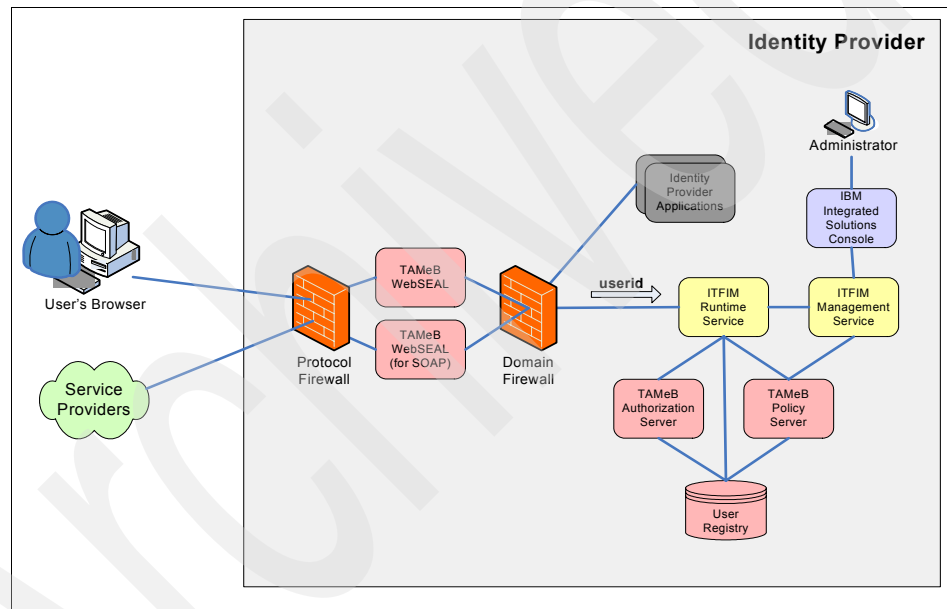


Figure 4-4 Base pattern for identity provider with SOAP Backchannel

4.1.3 Plug-in pattern

The Base pattern for Federated SSO can be modified to use the Access Manager for e-business Plug-ins rather than Access Manager for e-business WebSEAL as the *point-of-contact* server for an Tivoli Federated Identity Manager deployment. From a Tivoli Federated Identity Manager implementation perspective, there is little difference in using WebSEAL versus the plug-ins, as the required Access Manager for e-business functionality exists in both options.

In this design pattern, the Access Manager for e-business Web Plug-in is configured into the Web server acting as a proxy for the WebSphere Application Server hosting the Tivoli Federated Identity Manager services. Access Manager for e-business-based SSO will be provided to any application running in the same application server as Tivoli Federated Identity Manager. With Access Manager for e-business 5.1, SSO to applications running on application servers in the same DNS domain can be implemented, but it requires use of a domain cookie and a Access Manager for e-business plug-in must be installed in the Web server associated with the other application servers.

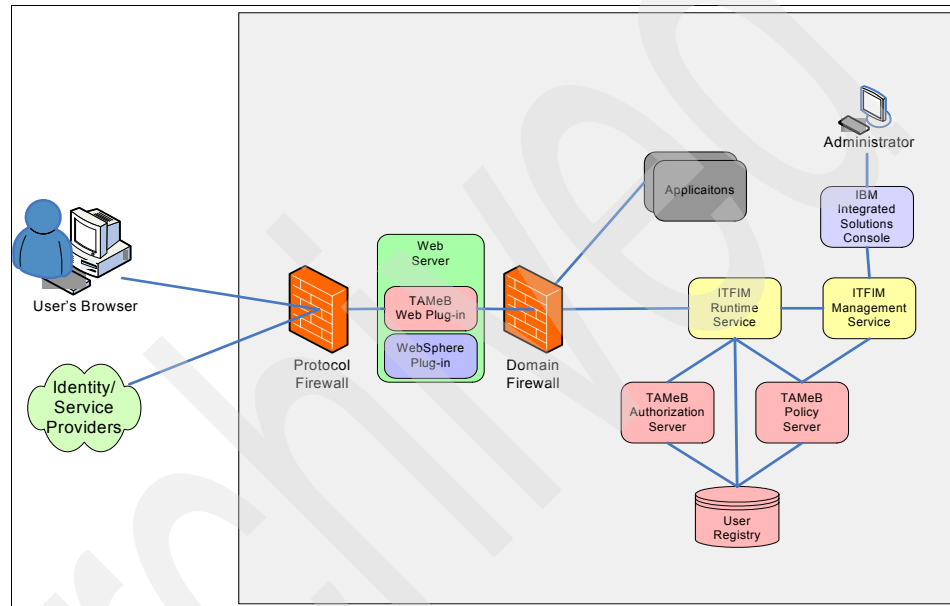


Figure 4-5 Plug-in pattern

Domain cookies are not generally considered ideal from a security perspective. Moreover, plug-in management can soon become problematic with even a small number of applications. So the Base pattern is recommended in all cases where Tivoli Federated Identity Manager will be used with more than one application.

4.1.4 Lightweight Access Manager for e-business pattern

In certain cases, Tivoli Federated Identity Manager can be deployed using a lightweight pattern for Federated SSO. In this pattern, Access Manager for e-business is leveraged for its session management capabilities *only*. Individual users are not stored in the Access Manager for e-business user registry and Access Manager for e-business related user management is largely done away with. Instead, the Access Manager for e-business user registry contains either a

single guest user ID or several role-based identities, with the identity mapping features of Tivoli Federated Identity Manager used to map to/from real user identities as required.

Since the standard Tivoli Federated Identity Manager Alias Service uses Access Manager for e-business UUIDs to identify which user is associated with a particular alias, the Lightweight Access Manager for e-business pattern cannot be used in cases where the standard Tivoli Federated Identity Manager Alias Service is being used. For example, standard Liberty account linking based Federated SSO cannot be used with this pattern; however, Liberty one-time use name identifier based Federated SSO can be deployed using this pattern.

For purposes of our discussion, we will base the description of the Lightweight Access Manager for e-business pattern on the Base pattern for Federated SSO, where Access Manager for e-business WebSEAL provides the point of contact services; however, the Plug-in pattern can also be adapted to use a lightweight Access Manager for e-business deployment in a similar manner. We will discuss the Lightweight Access Manager for e-business pattern from both the identity provider and service provider perspectives, but there is no requirement to use Tivoli Federated Identity Manager on both sides of the federation as part of this pattern. This pattern can be deployed independently on the identity provider or service provider side of a federation, or both sides if desired.

On the identity provider side, the key to the Lightweight Access Manager for e-business pattern is the use of the External Authentication Interface (EAI) feature of Access Manager for e-business; refer to Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, for a description of the interface.

Example

Figure 4-6 on page 145 illustrates a sample lightweight deployment of Tivoli Federated Identity Manager and Access Manager for e-business. In this lightweight deployment, a user is authenticated against an Enterprise directory but does not exist as an Access Manager for e-business user within the Access Manager for e-business registry. This is significant, because a user is (normally) required to exist within the Access Manager for e-business registry to allow Access Manager for e-business to build a local credential for the user. Recall that this credential is in turn used as part of the overall session management functionality provided by Access Manager for e-business and so this credential is an integral part of Access Manager for e-business functionality.

In this lightweight deployment, authentication is implemented through a custom login application. The Access Manager for e-business WebSEAL login page is redirected to this custom login application (Access Manager for e-business access control policy is defined such that this custom login application, and any

images uses, are accessible by unauthenticated users). The custom login application displays a login page to the user and validates the user credentials entered by the user, using whatever method is appropriate for the particular deployment. In our example, the custom login application validates the user ID and password entered by the user against a custom user registry. The custom login application is also responsible for handling any errors in login credentials entered by the user.

Once the login application has successfully authenticated the user, it sets several EAI-specific HTTP header variables on the reply to the user (via WebSEAL). WebSEAL intercepts the reply containing the EAI headers and uses the values of the HTTP header fields to create an Access Manager for e-business credential for the user. This Access Manager for e-business credential will be created for a guest user, and will include the user-specific information (user name, e-mail, and/or other attribute) as tag-value information. In our example, we pass the real user ID and associated e-mail address via HTTP headers from the custom login application.

When Tivoli Federated Identity Manager is invoked as part of the fulfillment of a single sign-on request, the user will be identified to Tivoli Federated Identity Manager as a guest user with these additional attributes. Tivoli Federated Identity Manager will then use an XSL rule to map these attributes from the (guest user based) Access Manager for e-business credential to the SAML assertion required for single sign-on.

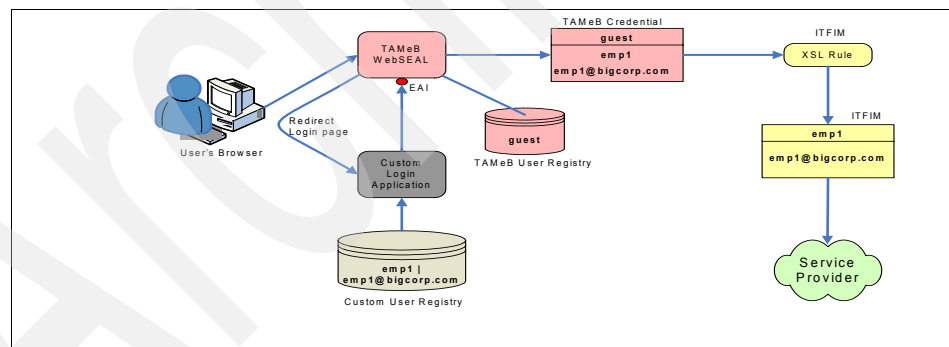


Figure 4-6 Example attribute flow for Lightweight pattern for identity provider

On the service provider side, the subject and attribute data contained in the incoming SAML assertion are used as input to setting HTTP header variables passed to the service provider applications, with Access Manager for e-business WebSEAL used as the link for passing this data from Tivoli Federated Identity Manager to the applications. The XSL rule used to map attributes from the incoming SAML assertion to Access Manager for e-business credential attributes is written such that it maps all users to a single guest user ID in Access Manager

for e-business. The Access Manager for e-business user registry only contains this guest user ID; it does not contain entries for each user identity that may be contained in an incoming SAML assertion.

In our example, we map the SAML subject and attribute to extended attributes in the Access Manager for e-business credential (via the XSL rule executed by Tivoli Federated Identity Manager). Access Manager for e-business WebSEAL is configured to pass these extended attributes to the back-end applications via HTTP header variables. Note that Access Manager for e-business allows different variables to be set for each junction.

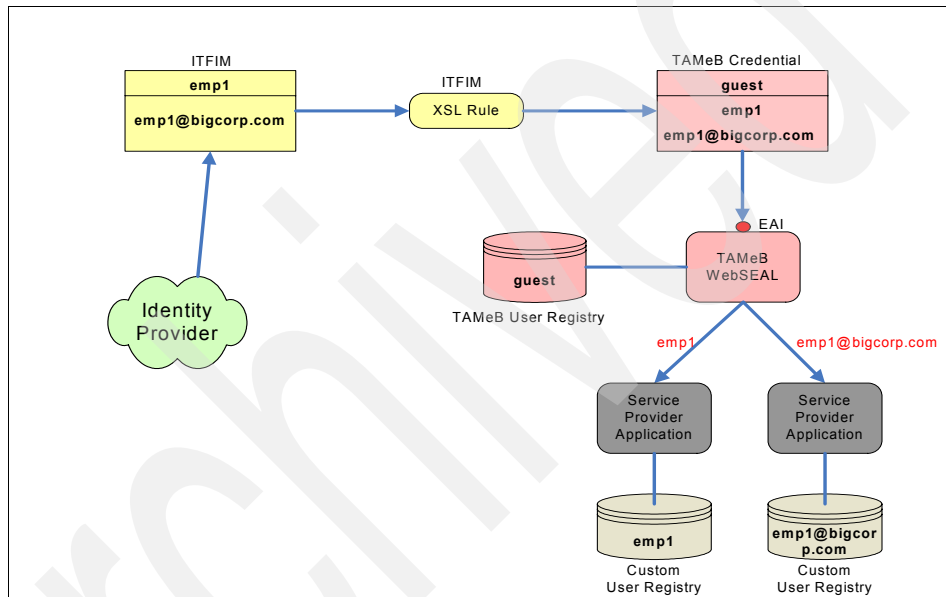


Figure 4-7 Example attribute flow for Lightweight pattern for service provider

This example could be extended to use a set of role-based identities in Access Manager for e-business, rather than a single guest user ID for all users. Logic would need to be added to the XSL rule, or Java code invoked from the XSL rule, in Tivoli Federated Identity Manager to implement the required mapping model. For example, instead of mapping all users to a single guest user ID, users can be mapped to one of many role-based identities, such as buyer, seller, agent, or manager, based on the attributes included in the single sign-on provided assertion.

We have described this architecture option as a separate pattern; however, it can co-exist with either the Base pattern or the Plug-in pattern.

4.1.5 Highly available architecture patterns

Any of the Federated SSO architecture patterns for Tivoli Federated Identity Manager described thus far can be extended for higher performance and availability via clustering techniques. Tivoli Federated Identity Manager fully supports a replicated Access Manager for e-business and Directory Server infrastructure. When replicated WebSEAL servers are part of a deployment architecture, Access Manager for e-business 5.1 requires an SSL-aware load balancer in front of these servers to load balance and provide fail-over for incoming requests. This load balancer needs to be configured to provide *sticky* sessions, such that all requests from a particular browser session will be routed to the same WebSEAL server instance. Multiple copies of the Tivoli Federated Identity Manager Management Console can be installed into an environment, and each console instance can manage multiple domains.

As a WebSphere Application Server based J2EE application, Tivoli Federated Identity Manager high availability is provided by clustering the underlying WebSphere Application Servers. When Tivoli Federated Identity Manager is deployed into a WebSphere Application Server (Version 6) cluster, the Tivoli Federated Identity Manager Management Service is installed into the Deployment Manager node. The Tivoli Federated Identity Manager Management Console is then used to deploy and remotely configure the Tivoli Federated Identity Manager Runtime applications into the managed nodes in the cluster. A set of Web servers is typically deployed between WebSEAL and the clustered WebSphere Application Servers to manage load balancing and failover.

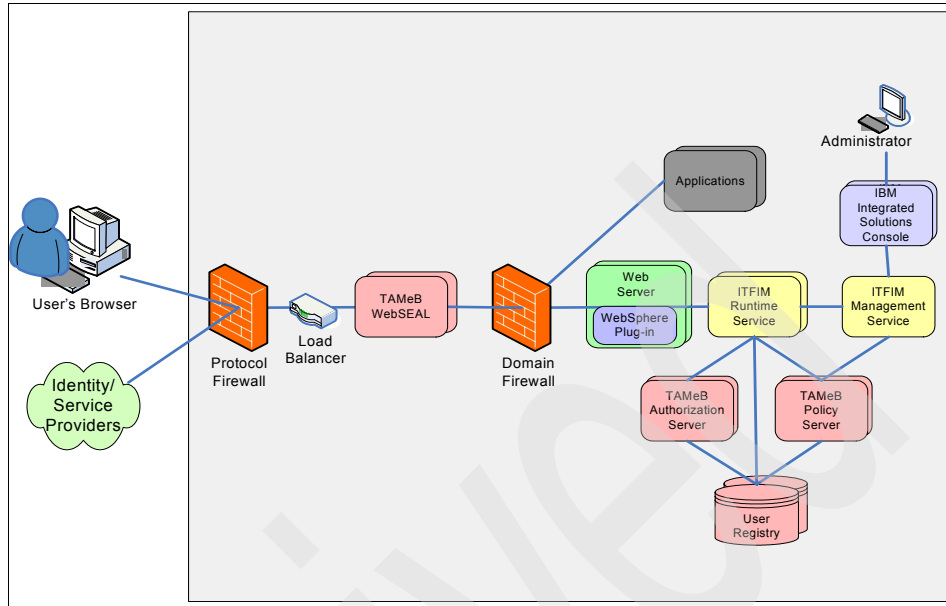


Figure 4-8 Clustered Base pattern

Tivoli Federated Identity Manager uses the shared configuration repository functionality of WebSphere Application Server 6 to manage its configuration data within a cluster. All changes made to the Tivoli Federated Identity Manager configuration using the Tivoli Federated Identity Manager Management Console are performed on the master configuration managed by the Tivoli Federated Identity Manager Management Service (running on the Deployment Manager node). Once all the changes are complete, the Tivoli Federated Identity Manager Management Console initiates a re-synchronization of the configuration repository data across all of the Tivoli Federated Identity Manager Runtime nodes in the cluster. Both clustered and non-clustered deployments of Tivoli Federated Identity Manager require the Tivoli Federated Identity Manager Runtime application to be stopped and restarted in order for the configuration changes to come into effect. In a clustered deployment, a *ripple restart* can be used to stop and restart each of the Tivoli Federated Identity Manager Runtime servers in turn, so as to keep the overall service available during the restart operation.

All Tivoli Federated Identity Manager Runtime nodes in a cluster use a shared session state, which is implemented using the DynaCache feature of WebSphere Application Server 6. This shared session state includes the assertion table for Browser Artifact profiles and contains sufficient information such that any of the nodes in the cluster can perform any operation. There is no need to ensure that subsequent operations for a particular Federated SSO session are directed to

the same instance of Tivoli Federated Identity Manager Runtime. For example, one Tivoli Federated Identity Manager Runtime node may perform a Federated SSO operation, but any of the nodes in the cluster have access to the session state information required to successfully perform a subsequent Single Logout operation for that session.

4.1.6 Multiple data center patterns

The clustered patterns for Federated SSO with Tivoli Federated Identity Manager can be further extended to include multiple, geographically distributed, data centers. Advice from senior WebSphere technical specialists indicates that it is not advisable to cluster WebSphere Application Server across a Wide Area Network (WAN), unless the throughput and latency of the link between the data centers is comparable to that provided by a Local Area Network (LAN). We therefore need to cater to the multiple data centers at the Tivoli Federated Identity Manager configuration layer.

The basic principle is that we configure each of the data centers as an independent identity/service provider in each federation they participate in. A WAN-based load balancing solution is required to handle load balancing and fail-over across the data centers. This WAN-based solution must be *sticky* in that it will send subsequent requests from the same browser session to the same data center.

The exact Tivoli Federated Identity Manager configuration details will differ depending on which Federated SSO protocol and associated profiles that you are using and whether you are hosting an identity provider or service provider. The different protocol solutions will run on the same Tivoli Federated Identity Manager infrastructure and may co-exist with other federations. It is only the federation configuration details that differ for each type of federation and role within the federation. All configuration and customizing of Tivoli Federated Identity Manager will need to be done independently at each data center; there is no shared configuration or session state across the data centers.

SAML 1.0

The configuration for SAML 1.0 depends on whether or not you are using the Browser Artifact profile for Federated SSO.

Browser Artifact Profile

If you are deploying the identity provider side of the Browser Artifact profile of SAML 1.0, we cannot solely rely on the *stickiness* of the WAN-based load balancing solution, as the Browser Artifact profile includes SOAP-based communication directly from the service provider to the identity provider. We therefore need to define a separate identity provider for each data center. The

configuration at each data center will use different provider ids and endpoint URLs even though they are logically performing the same role in the same federation. These provider id and URL endpoint values will use a logical host name that is unique to the data center. Service providers in the federation, regardless of whether they are implemented using Tivoli Federated Identity Manager, will define a distinct identity provider for each data center.

Requests initiated from the browser to the identity provider (for example, via a SSO link from a browser page) can use the logical host name that the WAN-based load balancing solution has been configured to balance across the data centers. The *stickiness* of the solution will ensure that subsequent requests after a Federated SSO operation will return to the same data center, and therefore be executed within the session state shared between the nodes at that data center. Note that with Access Manager for e-business 5.1 a WebSEAL server can only contain a single X.509 server certificate (which includes the logical host name), using the shared logical host name in some cases and the unique logical host name in other cases, will require a separate set of WebSEAL servers (junctioned to the same set of Tivoli Federated Identity Manager servers) at each data center for each logical host name so as to avoid browser warnings relating to an incorrect host name in the server certificate presented by the WebSEAL server.

With SAML 1.0, the service provider does not receive any inbound SOAP-based communication, so we can configure all of the data centers with the same configuration. The provider id and URL endpoints will use the logical host name that the WAN-based load balancing solution has been configured to balance across the data centers.

Browser POST Profile

If you are not using the Browser Artifact profile, you can either use the configuration described above for the Browser Artifact profile, or you can choose to use a simpler configuration.

Since the Browser POST profile of SAML 1.0 does not include any SOAP based communication between the identity provider and service provider, we can use the *stickiness* of the WAN-based load balancing solution to ensure that all (HTTP) requests initiated from, or redirected through, a particular browser session will be sent to the same data centers.

Under this scenario, the same configuration can be used at each data center for an identity provider or service provider. The provider id and URL endpoints will use the logical host name that the WAN-based load balancing solution has been configured to balance across the data centers. It is important here that the Tivoli Federated Identity Manager configuration at each data center contain the same provider IDs, endpoint URLs and signing keys, as the federation partners will be

configured to treat the multiple data centers as a single instance of the Identity provider or service provider.

WS-Federation

The WS-Federation (draft) standard does not currently contain any SOAP based communication between the identity provider and service provider. So we can therefore use the same approach described earlier for the SAML Browser POST profile, where the same configuration is defined at each data center.

Liberty ID-FF 1.1/1.2

The Liberty ID-FF standards contain a set of profiles with HTTP and SOAP based communication options for each of the operations defined in the standards.

If we restrict the profiles used to the HTTP based options, we can follow the same approach described earlier for SAML Browser POST profile, with the same configuration defined at each data center. SOAP based Liberty ID-FF profiles are not currently supported in the Multiple Data Center patterns with Tivoli Federated Identity Manager Version 6.

4.2 Federated Web services architecture patterns

Just as there are many different use cases for a single sign-on solution, there is more than one way to architect a Web services environment, especially one where security is taken into consideration. In this section, we discuss some of the typical deployment issues and architectures encountered with a Web services based approach to federation.

Technically, Tivoli Federated Identity Manager provides token validation, issuance (and exchange), identity mapping, and request authorization within a secure Web services environment. Tivoli Federated Identity Manager therefore supports scenarios such as those requiring the *normalization* of the security policy applied to a Web service. In this type of scenario, an application is deployed as a Web service with one security policy (the user must have the role of manager or the incoming request must include a SAML assertion), even though not all requestors will be able to satisfy this policy. Tivoli Federated Identity Manager can be used to provide the identity and attribute mapping required to determine the user's local roles based on those asserted by the requestor, so that the request includes the appropriate role of manager instead of Partner_Manager, for example. Similarly, Tivoli Federated Identity Manager can be used to provide token exchange functionality, so that a trust partner coming in over a VPN with a UsernameToken in the <Security> header can have their

request normalized to include the required SAML assertion, without requiring the partner to expand their capabilities to generate the required assertion.

4.2.1 Architecture approach

In this section we provide a quick review of the Tivoli Federated Identity Manager functionality leveraged within a Web services environment. We then go on to describe how to leverage this functionality in different scenarios.

The primary role played by Tivoli Federated Identity Manager in the architecture patterns for Federated Web services is to provide token validation, identity, and attribute mapping and/or authorization services to the XML gateways implementing WS-Security in the architecture. These services are invoked by the XML gateway using the WS-Trust interface of Tivoli Federated Identity Manager. The WS-Trust interface exposed by Tivoli Federated Identity Manager provides local access to the Tivoli Federated Identity Manager trust service, functionality referred to as the *security token Service (STS)*.

In addition to providing the trust service/security token service, Tivoli Federated Identity Manager Version 6 includes WebSphere Application Server specific components to provide the integration of WebSphere Application Server and Tivoli Federated Identity Manager. A WS-Trust client is provided with a WebSphere Application Server, to allow the WebSphere Application Server (through the WS-Security functionality) to invoke the Tivoli Federated Identity Manager Trust Service/Security Token Service. Tivoli Federated Identity Manager also includes a JAAS login module that allows a SAML assertion to be used to create a JAAS login context in a WebSphere Application Server. These WebSphere-specific components of Tivoli Federated Identity Manager that are related to Web services are collectively referred to as the Web services security management components of Tivoli Federated Identity Manager. In Tivoli Federated Identity Manager Version 6, Web services security management components are provided for WebSphere Application server Versions 5.1 and 6.0.

Token validation and exchange

Basically, the Tivoli Federated Identity Manager Security Token Service provides token validation and issuance functionality. Token validation is the process by which a token received at the STS is validated in terms of signatures on the token, expected structure, and contents of the token, and decryption of the encrypted contents (if any) of the token. Token issuance is the process by which a (new) token is created and returned to the (requesting) Trust Client by the security token service. Together, token validation and issuance can be used to implement *token exchange*. Token exchange allows for the validation of an incoming token type (such as a received SAML assertion) and the issuance of a

locally valid token (such as an Access Manager for e-business compatible credential, as is accomplished by the STS in a single sign-on scenario).

The incoming token (the token to be validated) is configured at the granularity of the partner making a request. This allows two different partners to request the same resource using different security tokens. The STS will handle the exchange of these received tokens for the token type required for application invocation.

Unlike the federated single sign-on environment, there is no one common, accepted (or required) token type associated with a Web service. SAML assertions are used in those situations where attributes about a requestor must be included in the request. Requests from a Java application client typically include a UsernameToken (an XML structure that includes a user name and a password). In those cases where the requestor has already determined the user's identity (there is no need to authenticate the user as the resource side) and no additional attributes (such as roles) are required, a simple IDAssertion (a UsernameToken that does not include a password) is often used to identify a requestor. A Kerberos ticket may be included as a BinarySecurityToken may be leveraged in a Microsoft Windows based rich client environment.

Web services resources may be deployed with one particular requirement on the expected incoming token type. Requestors may be able to include only a subset of possible token types in a Web services request. The Tivoli Federated Identity Manager TS/STS may be used to bridge this token type gap between requestors and resources.

Identity mapping

Just as identity mapping is used as part of federated single sign-on, there are requirements for identity mapping within a Web services environment. The attributes (identifiers, groups, roles, privileges, entitlements) used to identify a requestor in one environment may not match the attributes used within another environment. Rather than requiring a consolidation and normalization of internal attribute names across business partners, identity mapping functionality will allow locally valid attributes from one partner to be mapped to locally valid attributes at another partner, with no modifications to either partner's internal representation of these attributes.

Typically, a B2B or Web services environment is based on a transactional model, meaning that the Web services provider will honor an incoming transaction (provided it is correctly validated and trusted). This has the effect of removing the need for a one-to-one identity mapping within this environment. A user need not be identified as *Joe* at the Web services requestor. Because of the trust relationship between the requestor and provider, a many-to-one mapping may be used, so that Joe is mapped to PartnerXUser. Note that this does not mean that Joe's identifier is lost at the Web services provider side; it may still be included as

an attribute of the PartnerXUser, so that transactional verification allows actions by PartnerXUser and audit records can trace this user to Joe.

Tivoli Federated Identity Manager provides a flexible infrastructure for implementing the various identity mapping schemes found in Federated Web services.

Authorization

In a Web application server deployment, coarse-grained authorization of inbound HTTP(S) requests is increasingly being performed at the boundary to significantly reduce the number of unauthenticated requests entering an organizations network. Access Manager for e-business WebSEAL provides both the authorization decision and authorization enforcement functions for this boundary protection of Web-based operations.

A similar model can be applied to Federated Web services, with coarse-grained authorization performed at the boundary for incoming Web service requests. In this case, an XML gateway provides the authorization enforcement point, but Access Manager for e-business (via Tivoli Federated Identity Manager) can still be used as the authorization decision point. Tivoli Federated Identity Manager can optionally perform an Access Manager for e-business authorization API call to determine if the requesting user is authorized to access the service being requested. Since this is implemented in the Tivoli Federated Identity Manager trust service, the Access Manager for e-business call is transparent to the XML gateway and the requestor/provider applications. Any authorization failures result in the Web service request being rejected at the gateway and a SOAP fault returned to the requestor.

4.2.2 Point-to-point pattern

This pattern is included here for completeness, but it is not envisaged that this pattern will be used in many situations with Tivoli Federated Identity Manager Version 6.

Tivoli Federated Identity Manager version 6 does not include Web services security management support for outbound Web service requests from the WebSphere Application Server or WebSphere Application Client containers. So any token creation required at the client side of a Web services request would either need to be directly supported by the WebSphere Application container (which does not currently support SAML assertion tokens) or the Web service requestor would need to directly invoke the Tivoli Federated Identity Manager trust service to create the required token. The Tivoli Federated Identity Manager trust service provides a SOAP-based interface that implements the WS-Trust (draft) standard.

On the Web services provider side, Web services security management provides a WS-Trust client to allow incoming tokens to be validated and possibly exchanged for different tokens. This token exchange may also involve an identity mapping, where the identity in the incoming token is mapped, possibly on a many-to-one basis, to an identity relevant to the application being invoked. An Access Manager for e-business authorization call can also be configured to ensure the caller is authorized to invoke the request service.

If the token returned from the Tivoli Federated Identity Manager trust service is a SAML assertion, the Web services security management JAAS login module can be used to create a login context for the subject of the assertion and to make the assertion available to the application via the JAAS subject. The application can then access the JAAS subject value, parse the SAML assertion contained in the JAAS subject, and extract any additional attributes contained in the assertion. For example, the Web services provider application may use role-based identities from a WebSphere login perspective, but it may also require the real user's identity so it can be included in the audit logs. The Web services security management components of Tivoli Federated Identity Manager allow the incoming identity to be mapped to a role based identity, and for this role based identity to be used to create the login context in WebSphere Application Server. The original user's identity can be readily accessed by the application code, via the SAML assertion in the JAAS subject, so that it can be used in audit logging.

4.2.3 XML gateway pattern

The most common (and beneficial) use of Tivoli Federated Identity Manager Version 6.0 in Federated Web services deployments involves the use of an XML gateway (also sometimes referred to as an XML firewall or Web services gateway). The XML gateway is configured to invoke the Tivoli Federated Identity Manager trust service to validate and exchange security tokens. At a high level, one way to summarize the respective roles of the XML gateway and Tivoli Federated Identity Manager in this pattern is that the gateway implements WS-Security (and related standards) and Tivoli Federated Identity Manager implements WS-Trust.

Web services requestor

On the Web services requestor side, an XML gateway can be used as an outgoing proxy for Web services. The use of a gateway in this role allows the requestor applications to use security tokens and identities relevant to the local domain and ignore the complexities and differences involved in exchanging messages with partner organizations over an un-trusted network.

The Web services requestor side of the XML Gateway pattern for Federated Web services can be illustrated as follows.

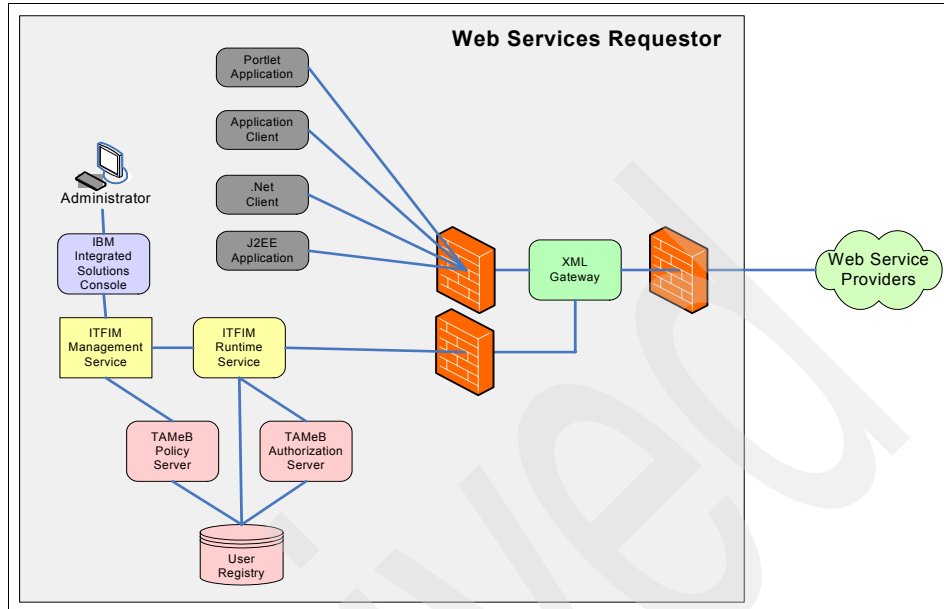


Figure 4-9 XML Gateway pattern for Web service Requestor

The requestor application sends a SOAP message containing a security token in a WS-Security header to the XML gateway. The gateway extracts the security token and sends a WS-Trust message to the Tivoli Federated Identity Manager trust service for token validation and exchange. The WS-Trust message includes the security token extracted from the header of the message, the identity of the calling application, and the identity of the target application. The Tivoli Federated Identity Manager server validates the token based on the configuration associated with the calling application, performs any specified identity mapping and Access Manager for e-business authorization calls, and generates a token applicable to the target application. The new security token is then returned to the gateway. The gateway replaces the security token in the message header, performs any other required message transformation and/or message level signing/encryption operations, and forwards the new message to the target service.

Web services provider

On the Web services provider side, the XML gateway fills the role of a reverse proxy for Web services. Again, this pattern allows the provider applications to use security tokens and identities relevant to the local domain and ignore the complexities and differences involved in exchanging messages with partner organizations over an un-trusted network

As mentioned earlier, a signed SAML assertion is a commonly used security token type for messages passing between organizations. The simplest form of security token that can be used to pass the user's identity to the Web services provider is an IDAssertion variant of a UsernameToken. It is envisaged that Kerberos based tokens will become increasingly popular in Microsoft Windows based environments in the future.

For those cases where attributes other than just the Subject from the incoming SAML assertion need to be passed to the provider application, the gateway can use a SAML assertion to pass both the subject and the additional attributes to the Web service provider application. You may choose to rely on the channel level security provided by SSL for this *internal* SAML assertion and leave it unsigned. As discussed in the Point-to-Point pattern earlier, the Web services security management JAAS login module can be used to create a login context for the subject of a SAML assertion received by a WebSphere Application Server and to make the assertion available to the provider application via the JAAS subject.

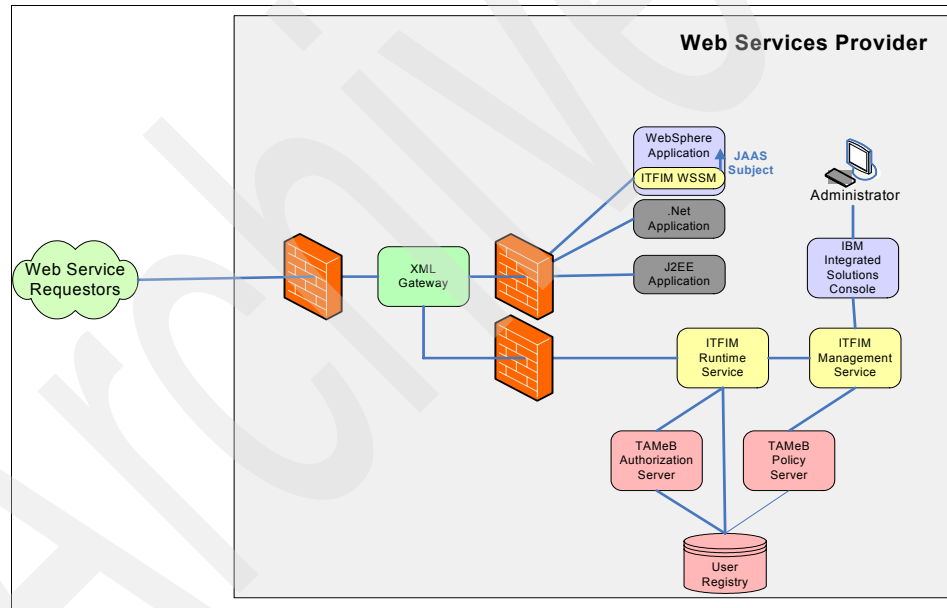


Figure 4-10 XML Gateway pattern for Web service provider

This pattern can be extended to include Access Manager for e-business WebSEAL in front of the XML gateway, with WebSEAL in the DMZ and the gateway moved inside the domain firewall. Motivation for doing this may include a desire to move the XML gateway from an outer DMZ to an inner DMZ or even into the protected segment of the network. This allows point-to-point security to Access Manager for e-business, so that Access Manager for e-business can

exclude any incoming requests that do not pass simple transport layer security requirements. This provides an extra layer of protection for the keys used to encrypt/decrypt and sign/validate messages while also providing an edge-level security layer.

Supported Gateways

The supported XML gateways for this pattern include any gateway that supports invoking a WS-Trust server, such as the Tivoli Federated Identity Manager trust service, for token validation and exchange.

Tivoli Federated Identity Manager ships with several (Web services security management) components that enable the IBM WebSphere Web services Gateway Version 6 to use the Tivoli Federated Identity Manager trust service in a manner consistent with this design pattern.

The following XML gateway/firewall vendors attended a WS-Trust interoperability event, sponsored by IBM in Austin during early May 2005, to test interoperability with the Tivoli Federated Identity Manager trust service and so have had their Trust Client implementations validated against the Tivoli Federated Identity Manager TS/STS:

- ▶ Datapower
- ▶ Layer 7 Technologies
- ▶ Reactivity
- ▶ Sarvega

4.3 Integrating applications into an F-SSO environment

Deployment of the Tivoli Federated Identity Manager functionality is not the same as integration of Tivoli Federated Identity Manager into an environment. Integration of Tivoli Federated Identity Manager requires an understanding of what applications are going to be exposed to federation users, what existing infrastructure can be reused to support this integration, and what customization is required to support the federation relationship.

4.3.1 Attribute flow between providers

As discussed earlier in this chapter, Tivoli Federated Identity Manager provides federated SSO to Access Manager for e-business, which in turn is responsible for providing SSO to applications. Access Manager for e-business may provide direct SSO to an application (or possibly the middleware on which it runs). As part of this Enterprise SSO solution, Access Manager for e-business may pass data via HTTP headers back to an application. When Tivoli Federated Identity Manager is integrated with an Access Manager for e-business solution, it

becomes possible for the two products to increase the scope of attribute flow, from point of contact to back end, to between partners to point of contact to back end.

As part of the integration of Tivoli Federated Identity Manager (and Access Manager for e-business) into an identity provider's environment, we must determine which (if any) attributes are to be provided to a service provider as part of an F-SSO solution. The following diagram illustrates the flow of attribute data from an identity provider implemented using Tivoli Federated Identity Manager.

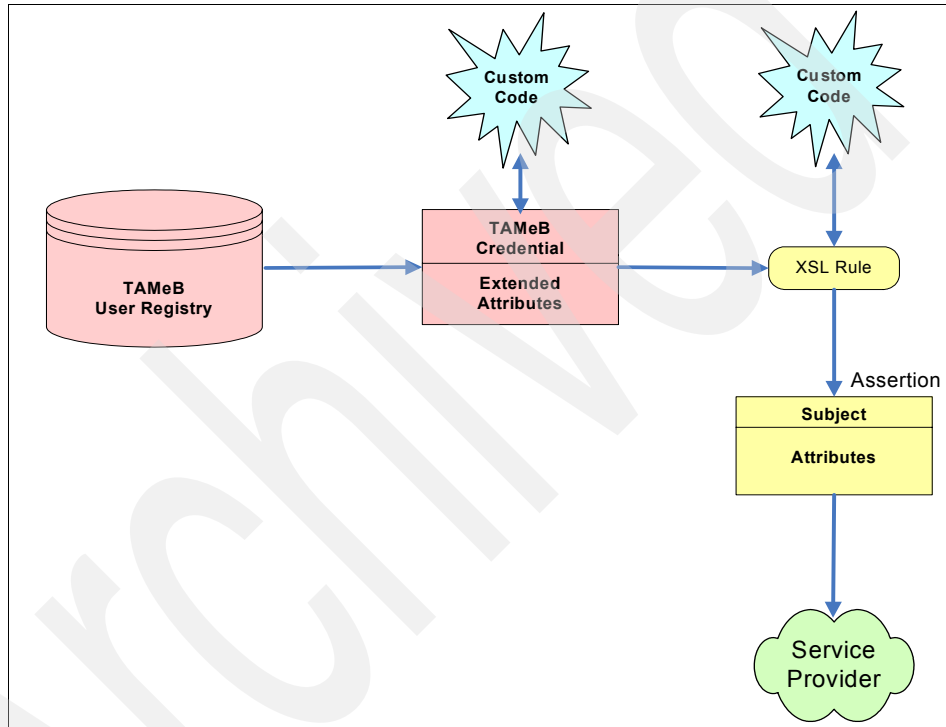


Figure 4-11 Attribute flow for identity provider

The first source of attributes to be included in a single sign-on assertion is from the Access Manager for e-business credential provided to Tivoli Federated Identity Manager to identify the user for single sign-on purposes. Attributes stored in an Access Manager for e-business credential are local attributes retrieved from the Access Manager for e-business registry during credential creation (part of the authentication process). Additional attributes are stored as extended attributes in the Access Manager for e-business credential for the user. Access Manager for e-business also provides an interface that allows custom C code to be written to provide additional extended attributes to be stored in the

Access Manager for e-business credential. This custom code is executed when the Access Manager for e-business credential is created by WebSEAL.

Once the Access Manager for e-business credential has been created, Tivoli Federated Identity Manager uses an XML version of the Access Manager for e-business credential as input to the identity/attribute mapping step performed as part of the assertion generation. This mapping is defined by an XSL rule. This mapping may include a simple copy of the existing (credential defined) attributes, a mapping of attributes from one value to another, or the retrieval of additional attributes. Custom Java modules can be invoked from these XSL rules to obtain additional attribute data that is not available in the Access Manager for e-business credential. These XSL rules and any associated Java modules are invoked for every assertion generated by the identity provider and are specific to the identity provider-service provider relationship.

On the service provider side, the flow of attribute data from an incoming assertion to a service provider application is illustrated in the following diagram.

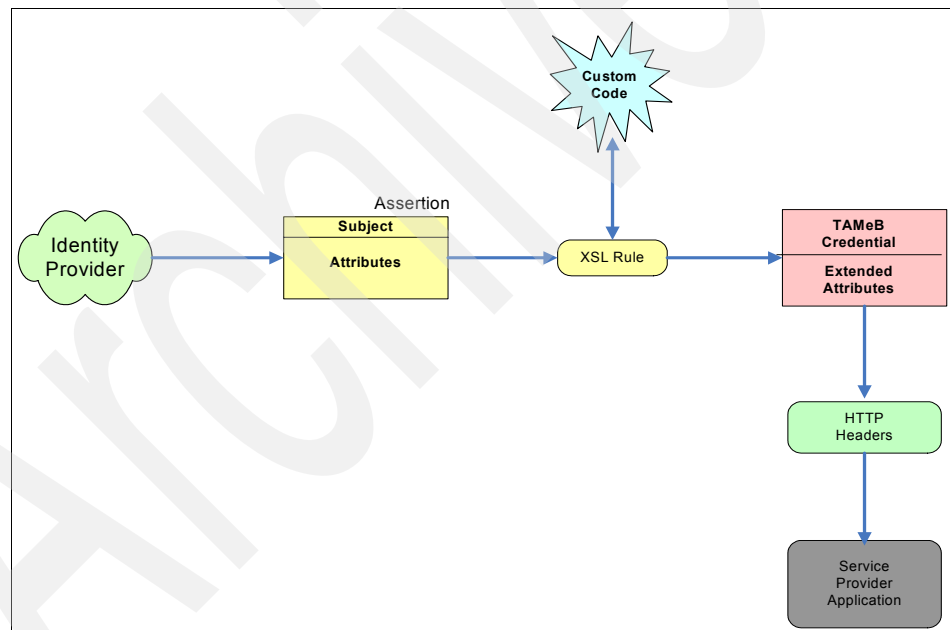


Figure 4-12 Attribute flow for service provider

In this scenario, the single sign-on assertion received at the service provider may contain attributes about a user. These attributes (and the information contained in the assertion) are translated into a Tivoli Federated Identity Manager internal format and an XSL rule is used to map this information and then format it as an Access Manager for e-business credential. This mapping may include a simple

copy of the existing (assertion defined) attributes, a mapping of attributes from one value to another, or the retrieval of additional attributes. Custom Java modules can be invoked from these XSL rules to obtain additional attribute data that is not available in single sign-on assertion. These XSL rules and any associated Java modules are invoked for every assertion received at the service provider and are specific to the identity provider-service provider relationship.

Access Manager for e-business WebSEAL can then be configured to extract particular attributes from the Access Manager for e-business credential and send the attribute values to the service provider applications via HTTP header variables. Access Manager for e-business WebSEAL allows different attributes to be sent to different applications.

4.3.2 User-controlled federated life cycle management

Application developers may choose to add Federated SSO links to their pages to customize a user's federation experience. These links may provide account linking/delinking, single logout, SSO to other applications and/or other operations supported by the associated protocol.

This can point to the specific page template customization of the next section, or they can be collapsed into a single section.

4.3.3 Customized user-managed federation management

Tivoli Federated Identity Manager includes an *Info Service API* for querying the Tivoli Federated Identity Manager Management Service for federation-related data. The Info Service API allows an application to determine if a user's account is currently linked to an account at a specific partner. This feature can be used to dynamically build a page showing a list of links to partner sites for which the current user already has an account linked to their local account, and possibly provide a separate list of links that would allow the user to link their account to specific partner sites with which their local account is not currently linked.

Thus a user can be provided with a listing of *Partners you have federated with (single sign-on partners)* and a separate listing of *Partners you haven't federated with*. A related example is shown in Figure 4-13 on page 162.

Welcome Alison, to your MyPhone Page

<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">Linked Services</td></tr> <tr style="background-color: #ffff00;"><td style="text-align: center;">First Phone Mobility</td></tr> <tr style="background-color: #ff69b4;"><td style="text-align: center;">Messaging Center</td></tr> <tr style="background-color: #add8e6;"><td style="text-align: center;">First Internet</td></tr> <tr><td style="text-align: center;">. . .</td></tr> </table>	Linked Services	First Phone Mobility	Messaging Center	First Internet	. . .	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">Personal Profile Information</td></tr> <tr><td style="padding: 5px;">View, change information about yourself, such as your email address, mailing address, contact preferences, and so on</td></tr> </table>	Personal Profile Information	View, change information about yourself, such as your email address, mailing address, contact preferences, and so on
Linked Services								
First Phone Mobility								
Messaging Center								
First Internet								
. . .								
Personal Profile Information								
View, change information about yourself, such as your email address, mailing address, contact preferences, and so on								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">Available Services</td></tr> <tr style="background-color: #ffff00;"><td style="text-align: center;">Business Solutions</td></tr> <tr style="background-color: #ff69b4;"><td style="text-align: center;">Satellite TV</td></tr> <tr><td style="text-align: center;">. . .</td></tr> </table>	Available Services	Business Solutions	Satellite TV	. . .	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">Create Service Requests</td></tr> <tr><td style="padding: 5px;">Moving? Need repair services? Create a service request online!</td></tr> </table>	Create Service Requests	Moving? Need repair services? Create a service request online!	
Available Services								
Business Solutions								
Satellite TV								
. . .								
Create Service Requests								
Moving? Need repair services? Create a service request online!								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="text-align: center;">Other Links</td></tr> <tr style="background-color: #ffff00;"><td style="text-align: center;">Do Not Call Registry</td></tr> <tr style="background-color: #ff69b4;"><td style="text-align: center;">Yellow Pages</td></tr> <tr style="background-color: #add8e6;"><td style="text-align: center;">. . .</td></tr> </table>	Other Links	Do Not Call Registry	Yellow Pages	. . .				
Other Links								
Do Not Call Registry								
Yellow Pages								
. . .								
<p>First Phone Stock Price: \$10.31 at 9:42am (20 min delayed)</p>								

Figure 4-13 Linked services for user Alison

The Tivoli Federated Identity Manager Info Service API can also allow an application program/portal to obtain the URLs for specific Federated SSO operations for specific partners. This allows an application developer to avoid placing hard-coded links to Tivoli Federated Identity Manager functionality on their pages.

4.4 Customizing F-SSO

This section describes how Tivoli Federated Identity Manager can be customized to provide the look and feel required for a particular deployment through the (HTML) page templates provided with Tivoli Federated Identity Manager.

4.4.1 Customizing page templates

Tivoli Federated Identity Manager ships with a set of page templates for:

- ▶ Consent to Federate page
- ▶ Where Are You From page
- ▶ Automatic POST pages
- ▶ Operation success pages
- ▶ Error pages.

These default page templates can be customized to fit the requirements of a particular deployment. The page templates contain various macro variables that Tivoli Federated Identity Manager will replace with the corresponding value as it builds a page.

The Tivoli Federated Identity Manager configuration file `sps.xml` contains the mapping from logical page name to physical page. In some cases you may need to modify an entry in `sps.xml` to customize a page for a specific event, as many of the error events are mapped to generic error pages.

In some cases, customizing specific Tivoli Federated Identity Manager error pages may provide an opportunity to provide error recovery from a user experience perspective. For example, the default error page that is displayed when a user attempts to perform a Liberty ID-FF SSO operation and their account has not been linked to an identity provider account contains an error message and a stack trace. This page can be easily customized to inform the user that their account is not yet linked to an identity provider account and to provide an option to allow the user to initiate an account linking operation.

At the time of writing this Redbook, the error event entries in `sps.xml` and the associated page templates and macro variables had not yet been documented in the product manuals. This implies that any customization is likely to involve some careful trial and error and is not likely to be officially supported.

4.4.2 Customizing Access Manager for e-business page templates

Access Manager for e-business also ships with a set of page templates. The Access Manager for e-business product documentation describes how these templates can be customized.

Additional customization for Access Manager for e-business pages in an Tivoli Federated Identity Manager environment might include:

- ▶ Adding Federated SSO links to the Access Manager for e-business login page on a service provider.
- ▶ Modifying the Access Manager for e-business login page on an Identity provider to include the purpose of the authentication being requested (for example, to access to a local protected resource, to SSO to another site, or to identify an account to be linked to the service provider account).

4.4.3 Storing aliases

By default, the standard Tivoli Federated Identity Manager Alias Service module stores aliases (also know as Name Identifiers) used in the Liberty ID-FF protocols under the root LDAP suffix `cn=itfim`. This location in the LDAP tree can be modified prior to creating any aliases by modifying the alias root in the Alias Service configuration file `lids.xml`.

If you intend to run more than one instance of Tivoli Federated Identity Manager on a single machine, the alias root suffix values should be made to be unique for each instance. For example, if you are setting up a simple test system for Federated SSO, you may choose to store the aliases from one instance under `cn=idp,cn=itfim` and the aliases for the other instance under `cn=sp,cn=itfim`.

The Tivoli Federated Identity Manager Alias Service is designed to be a pluggable interface. A DB2-based Alias Service is available for those customers who want to use Liberty ID-FF with very large numbers of users.

4.5 Solution design considerations

This section contains a series of short discussions on topics relating to designing a solution for deploying Tivoli Federated Identity Manager in a real-world environment. This information was mostly collated during early deployments of Tivoli Federated Identity Manager in the Early Support Program.

4.5.1 Exchanging metadata with your partners

Once the business and legal agreements are in place, you will define the attributes of your role in the federation using the Tivoli Federated Identity Manager Management Console and then share that *metadata* with your partner(s).

The technical information to be shared and agreed upon with your partner(s) for Federated SSO includes:

- ▶ Federated SSO protocol and version to be used
- ▶ Provider ID (or Realm, depending on the protocol you are using)
- ▶ Profiles within the protocol to be supported
- ▶ Endpoint URLs for each of the profiles to be supported
- ▶ Public certificates for validating your digital signatures
- ▶ CA certificate for the server certificate in your *point of contact* server
- ▶ Method for client authentication of the SOAP connections (none, X.509 certificate), plus the CA certificate and Distinguished Name (DN) of the client certificate if needed
- ▶ Type, value range and semantics of the Subject field in the assertion
- ▶ Name, type, value range and semantics of any attributes to be included in the assertion
- ▶ Session timeouts and request/assertion lifetimes.

Some Federated SSO protocols, for example the Liberty ID-FF protocols, include a definition of a metadata format for exchanging some of this data. Where the protocol defines a metadata format, you can use the Tivoli Federated Identity Manager Management Console to export your metadata and import that of your partners.

4.5.2 Availability of IBM Access Manager for e-business policy server

In a standard Access Manager for e-business deployment, all of the servers, with the exception of the Policy Server, can be replicated for load balancing and fail-over. The best practice for deploying the Access Manager for e-business Policy Server is to create a *warm standby* Policy Server that can be activated in the event that the Policy Server is unavailable for an extended period.

In an Access Manager for e-business deployment without Tivoli Federated Identity Manager, all run-time operations will continue to operate if the Policy Server is unavailable. However, the Tivoli Federated Identity Manager servers use the Access Manager for e-business Administration API to terminate user sessions in Access Manager for e-business WebSEAL servers during Single Logout operations. The Access Manager for e-business Administration API relies on the Access Manager for e-business Policy Server to act as an intermediary for communication with the WebSEAL servers. So the requirement for keeping the Access Manager for e-business Policy Server available is stronger when Tivoli Federated Identity Manager is deployed.

At the time of writing this Redbook, a developerWorks article is being written by Tivoli development to describe a high-availability architecture for Access

Manager for e-business that includes replicated *read-only* Policy Servers suitable for use with Tivoli Federated Identity Manager.

4.5.3 Key management

Federated SSO protocols make use of a number of digital keys to sign requests and validate signatures on responses. Similarly, the SAML assertions used in Federated Web services are typically signed. It is important to note that digital signing and validation operations will fail if the key being used has expired. As many of the keys obtained from public Certificate Authorities have a lifetime of 12 to 24 months, it is important to establish a manual procedure to proactively replace keys before they expire. It is also important to monitor your partner's keys and advise them if their keys are nearing expiry.

The Tivoli Federated Identity Manager Management Console provides support for reviewing expiry dates on signing/validation keys.

4.5.4 Session timeout

A key issue to consider in designing a Federated SSO solution is session timeout (either due to session duration or session inactivity). The Federated SSO standards bodies have not yet addressed this issue. From a user perspective, the ideal solution would be to present the appropriate identity provider login page as required after session duration/inactivity timeout.

Depending on the nature of the federations defined, it may be possible to add some JavaScript to the service provider login page to automatically initiate a Federated SSO operation on session timeout; otherwise the user will have to choose to initiate the Federated SSO operation from the links shown on the service provider login page.

A related requirement that may be raised in Federated SSO environments is to link the inactivity timers for the identity provider and service providers, such that while a user is using a particular service provider resource, the associated identity provider session will remain active. One situation where this requirement is important is where a service provider site is being accessed in an iFrame portlet on an identity provider hosted portal. In this case, a user may find it disconcerting to be required to re-authenticate due to activity when they press a link in the surrounding portal page after having just been working inside the service provider portlet on the same page.

One solution to this requirement that will work regardless of which vendor's products are used at the identity provider and service provider, is to have a (possibly hidden) image from the identity provider site on every service provider application page. This image may possibly be incorporated into the page design

to highlight the source of the authentication. Alternatively, a servlet filter may be added to the service provider application(s) to add a hidden image to each page returned to the browser.

4.5.5 Application logout

Another key issue to consider in designing a Federated SSO solution is application logout. Protocols such as Liberty ID-FF and WS-Federation include profiles for Single Logout (SLO). An SLO operation will terminate the user session at the identity provider as well as terminating any service provider sessions that used that identity provider session for authentication. The motivation for SLO lies in the belief that if a user is transparently logged into multiple sites from a single authentication, then a similar model should be used for logout.

This is an amiable goal, but there are several problems with the implementation. Many of the SLO profiles in the standard Federated SSO protocols rely on the user to inspect the logout success/failure messages coming from different products (with different customization) to determine the overall success/failure of the SLO operation. Moreover, if a user is unaware of the Federated SSO being performed between various sites, they may have trouble understanding why they are being presented with a list of logout success/failure messages. At a minimum, it is recommended that the SLO failure messages be modified to advise the user to close all browser sessions to ensure the user is fully logged out. You may also consider adding similar advice to the SLO success pages to inform the user that it is safe practice to close all browser sessions to ensure successful logout across all sessions.

In a Tivoli Federated Identity Manager deployment (at either the identity provider or service provider), termination of the current user session at the local node is effected using the Access Manager for e-business Administration API to terminate the session in the session cache of WebSEAL (or the Access Manager for e-business Web plug-in). Success or failure is determined by the return code from this API. In a standard Access Manager for e-business deployment (without Tivoli Federated Identity Manager), it is accepted best practice to add some JavaScript to the Access Manager for e-business logout success (and failure) pages to delete all session cookies associated with the applications protected by Access Manager for e-business. By default, Access Manager for e-business renames all cookies coming from *junctioned* applications to avoid accidental overwriting of cookies with the same name from different back-end servers. The following JavaScript function illustrates how the cookies from the back-end applications can be identified and *deleted*. If you call this function as your page is loading, it will delete all cookies from applications junctioned behind WebSEAL.

```
<script language=javascript type="text/javascript">
function deleteJunctionCookies() {
```

```

var TAMPrefix = "AMWEBJCT!";
var cookies = "" + document.cookie;
var cookieArray = cookies.split("; ");
for (var i = 0; i < cookieArray.length; ++ i) {
    var firstCh = cookieArray[i].indexOf(TAMPrefix);
    if (firstCh == 0) {
        var length = cookieArray[i].indexOf("=");
        var name = cookieArray[i].substr(firstCh, length);
        document.cookie = name + "="; expires=Fri, 03-Dec-1993 04:10:00
CET; path="/";
    }
}
}
}
</script>

```

A similar technique can be used in an Tivoli Federated Identity Manager environment for HTTP-based SLO profiles. Javascript to delete cookies for back-end servers can be added to SLO success (and failure) page templates used by Tivoli Federated Identity Manager. However, the Liberty ID-FF standards include SOAP based profiles for SLO. With these SOAP based profiles, the partner nodes do not have an opportunity to run any JavaScript on the browser to delete the application cookies. It is therefore recommended that in an Tivoli Federated Identity Manager environment using SOAP-based SLO profiles, the Access Manager for e-business login page also be updated to include some JavaScript code to delete the back-end application cookies.

This technique for deleting cookies with SOAP-based SLO profiles does not address all threat scenarios, so it is also recommended that applications in this environment verify incoming requests to ensure that the value of the HTTP Header variable in the request, which contains the user identity from Access Manager for e-business, matches the local user login context. For standard Access Manager for e-business SSO configurations, this HTTP Header variable would be iv-user; however, in an Tivoli Federated Identity Manager environment the real user identity may be passed to the application via a different HTTP Header variable.

Of course, closing all browser sessions on logout removes all risks associated with unexpired application session cookies.

4.6 Conclusion

This chapter described architecture options for deploying Tivoli Federated Identity Manager, and approaches for integrating this software product with other middleware and customer applications. Architecture pattern for Federated SSO and for Tivoli Federated Identity Manager Federated Web services were

introduced. Then it was shown how to integrate applications into a Tivoli Federated Identity Manager F-SSO environment, and how to customize Tivoli Federated Identity Manager for F-SSO. Finally, a series of short discussions on topics relating to designing a solution for deploying Tivoli Federated Identity Manager in a real-world environment completed the chapter.

Archived

Archived



Integrating with IBM identity management offerings

Federated identity management (FIM) is an administration concept. It enables companies to extend an organization's identity management infrastructure to their business partners. As such, IBM's Tivoli FIM solution will extend identity management for both the identity provider and service provider infrastructure. Tivoli Federated Identity Manager extends the current Tivoli identity and security offerings: Tivoli Identity Manager, Tivoli Access Manager Family, Tivoli Directory Server, and Tivoli Directory Integrator. This chapter briefly describes some of these offerings.¹

¹ This chapter content is adapted from the *IBM Federated Identity Management* white paper (Heather Hinton, et al).

5.1 IBM Tivoli Access Manager for e-business

Tivoli Access Manager provides authentication, authorization and session management services. Access Manager (WebSEAL) provides a centralized session management service for Web (HTTPS) and SOAP Web services for user-based and serviced-based transactions. Access Manager provides a Policy Decision Point (PDP) with its authorization server and policy server. Access Manager provides several out-of-the-box Policy Enforcement Points (PEP), primarily the HTTP-based WebSEAL reverse proxy and Web plug-in, and provides the ability to implement customized PEPs through its industry-standard Java and C API.

Access Manager provides authentication services where the authentication process is a direct interaction with the end user (such as a traditional user name/password challenge response) or a proprietary Access Manager-based cross-domain single-sign-on solution. In a federated scenario, Access Manager establishes and controls a session for a user in response to a federation-based interaction.

IBM's federated identity management solutions extend Access Manager authentication and session management functionality by providing standards and public specification-based single-sign-on and federation user session life cycle solutions.

5.1.1 Identity provider integration

In general, Access Manager will treat Tivoli Federated Identity Manager as a generic back-end application that does not have explicit integration requirements with FIM. When an enterprise is configured for *identity provider* functionality, Access Manager will:

- ▶ Provide session management services for local users, including:
 - Authentication services for local users, providing support for direct authentication of users
 - Authorization services, providing access control for local resources based on local access policy
- ▶ Provide access control to FIM functionality.

Provide authorization/access to FIM federated single-sign-on solutions. Access Manager authorization decisions can be used to provide fine granular access to FIM, so that only properly authorized users are able to participate in a federated solution.

When configured for identity provider functionality, IBM's FIM solutions integrate with Access Manager and have expectations on Access Manager behavior as follows:

- ▶ Tivoli Federated Identity Manager will be configured as a normal back-end resource to provide federated single sign-on functionality, including SSO, single sign-off, account linking, and so on.
- ▶ Tivoli Federated Identity Manager will rely on Access Manager as an authorization service to ensure that only authorized users are able to invoke FIM solutions.
- ▶ Tivoli Federated Identity Manager will rely on Access Manager to identify users for purposes of federation functionality based on Access Manager asserted information (for example, `iv_user`, `iv_creds`).

5.1.2 Service provider integration

When an Enterprise is configured for *service provider* functionality, Access Manager will:

- ▶ Provide session management services for (local and federation) users, including:
 - Authentication services for local users, providing support for direct authentication of these users
 - Session establishment for federation users, based on federated single sign-on functionality implemented by FIM
 - Authorization services, providing access control for all users to local resources based on local access policy
- ▶ Provide access control to FIM functionality:
 - Treat FIM single sign-on functionality as a publicly available resource
 - Treat FIM session life cycle functionality (such as session logout) as a protected resource accessible only to authenticated/authorized users

When configured for service provider functionality, IBM's FIM solutions integrate with Access Manager and have expectations on Access Manager behavior as follows:

- ▶ FIM will be configured as a “normal” back-end resource to provide federated single sign-on functionality, including SSO, SSOFF, account linking, and so on.
- ▶ FIM will rely on Access Manager as an authorization service to ensure that only authorized users are able to invoke FIM solutions.
- ▶ FIM will provide Access Manager with information to build a session for a user in response to a successful SSO by the user.

- ▶ FIM will rely on Access Manager to identify users for purposes of (non-SSO) federation functionality based on Access Manager asserted information (for example, iv_user, iv_creds).

You can find more information about how to configure Tivoli Access Manager in Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363.

5.2 IBM Tivoli Identity Manager

Tivoli Identity Manager provides Enterprise-wide identity management and user provisioning functionality. It integrates with many different types of systems, including operating systems, databases, directories, and ERP solutions such as PeopleSoft and SAP. This allows Human Resources administrators to manage users through a dedicated system while having Identity Manager manage the ongoing creation, change, and deletion of user accounts. Identity Manager provides a workflow engine to automate the business process of user management as well as a set of Java APIs to simplify integration with homegrown applications. Tivoli Federated Identity Manager extends Identity Manager's enterprise provisioning capability with federated provisioning. Federated provisioning extends the concept of automated user provisioning to trusted third-party organizations such as suppliers, business partners, and service providers. Federated provisioning can also help extend enterprise provisioning solutions to support intranet organizations such as autonomous regional organizations that have a need to manage user provisioning locally. Tivoli Federated Identity Manager can leverage Identity Manager to implement the local provisioning of a user in response to a federated provisioning request. This allows a local Enterprise to maintain locally relevant user information, including user life cycle functionality, for a federated user, in response to provisioning information from federation business partners.

Tivoli Federated Identity Manager extends Identity Manager's provisioning and workflow functionality by providing standards and public specification-based federated provisioning for user life cycle management. Providing this support through Tivoli Federated Identity Manager provides a modular solution that allows easy extensibility of federation and provisioning solutions with minimal impact on an existing Identity Manager environment.

5.2.1 Identity provider integration

In general, Identity Manager will treat FIM as a provisioning endpoint. When an Enterprise is configured for *identity provider* functionality, Identity Manager will provide local identity management, including workflow and provisioning.

- ▶ Apply workflow functionality to the overall management of users within local enterprise, including initiating a possible workflow/approval process (if required) to authorize provisioning to a FIM endpoint.
- ▶ Provide provisioning solutions to push/create user information at required local endpoints such as application-specific user repositories.

When configured for identity provider functionality, IBM's FIM solutions integrate with Identity Manager and have expectations on Identity Manager behavior as follows:

- ▶ FIM will rely on Identity Manager as the authoritative source for information that is to be provisioned to federation business partners.
- ▶ FIM will act as a local Identity Manager endpoint for provisioning purposes; FIM will initiate federated provisioning in response to a Identity Manager provisioning request.
- ▶ FIM will receive notifications and provisioning responses from federation business partners and will “proxy” this information to Identity Manager.

5.2.2 Service provider integration

In general, Identity Manager will treat Tivoli Federated Identity Manager as a provisioning source. When an Enterprise is configured for *service provider* functionality, Identity Manager will provide local identity management, including workflow and provisioning.

- ▶ Apply workflow functionality to the overall management of users within the local enterprise, including initiating a possible workflow/approval process (if required) to authorize local provisioning based on a FIM provisioning trigger.
- ▶ Provide provisioning solutions to push/create user information at required local endpoints such as application-specific user repositories in response to a FIM provisioning trigger.

When configured for service provider functionality, IBM's Tivoli Federated Identity Manager solutions integrate with Identity Manager and have expectations on Identity Manager behavior as follows:

- ▶ FIM may also provision information to a local repository that is in turn monitored by Identity Manager to trigger a local Identity Manager provisioning event.

- ▶ FIM will proxy any Identity Manager provisioning responses (such as status and notification) as required.
- ▶ FIM will appropriately respond to the identity provider federated provisioning functionality with status/notification, based on the information received from the local Identity Manager.

In 3.2.7, “Provisioning services” on page 98, we discussed the Web services provisioning service.

5.3 IBM Tivoli Directory Integrator

IBM Tivoli Directory Integrator provides a lightweight data synchronization solution. This allows a simple solution for keeping multiple data stores in synchronization, even when there is no one single *authoritative* data store.

Tivoli Federated Identity Manager internalizes Directory Integrator functionality within its federated provisioning solution. As such, the integration required with Directory Integrator is part of the installation and configuration of Tivoli Federated Identity Manager itself.

Tivoli Federated Identity Manager extends Directory Integrator's data synchronization solutions to provide standards and public specification-based federated provisioning and single sign-on. By leveraging Directory Integrator FIM is able to provide a modular solution that allows easy extensibility of federation and provisioning solutions with minimal impact on an existing enterprise environment.

5.3.1 Identity provider integration

Directory Integrator functionality synchronizes local identity provider data stores

FIM/Directory Integrator functionality:

- ▶ In response to events within monitored data stores, builds a (WS-Provisioning) federated provisioning request
- ▶ As part of building a federated provisioning request, implements markup language translation (for example, DSMLv2 to DAML), if required
- ▶ Acts as a client to the Tivoli Federated Identity Manager federated provisioning service

5.3.2 Service provider integration

Directory Integrator functionality synchronizes local service provider data stores.

FIM/Directory Integrator functionality:

- ▶ Receives (WS-Provisioning) federated provisioning requests
- ▶ As part of building a local provisioning request, implements markup language translation (for example, DSMLv2 to DAML), if required
- ▶ Receives federated provisioning events, validates them, and triggers local provisioning to a local data store that is monitored by the local Identity Manager
- ▶ Receives federated provisioning events, validates them, and directly triggers a provisioning event (including workflow) at a local Identity Manager

In 3.2.7, “Provisioning services” on page 98, we discussed the Web services provisioning service, and how IBM Directory Intergrator functionality is exploited with the Tivoli Federated Identity Manager solution.

5.4 IBM Tivoli Directory Server

Tivoli Directory Server provides a highly available, scalable LDAP directory that can act as an enterprise’s main data repository. Tivoli Federated Identity Manager will leverage a data store (such as Directory Server) for the management of internal (relevant to Tivoli Federated Identity Manager only) information. Tivoli Federated Identity Manager may also require integration with a local data store as part of the fulfillment of federation functionality.

By leveraging Directory Server, Tivoli Federated Identity Manager is able to provide modular, highly scalable solutions that allow extensibility of federation and provisioning solutions with minimal impact on an existing enterprise environment.

5.4.1 Identity provider integration

Some information on identity provider integration:

- ▶ Store common identifiers that are used when communicating with a service provider/business partner about a given local user.
- ▶ Integrate with a local data store to retrieve information about users as part of building a single sign-on response (to a SSO request issued by a service provider).

5.4.2 Service provider integration

For service provider integration:

- ▶ Store common identifiers that are used when communicating with an identity provider/business partner about a given local user.
- ▶ Store identity information about a user that can be used to build a local session in response to a single sign-on response from an identity provider.

5.5 IBM WebSphere Application Server

The IBM middleware platform for Java and Web services is WebSphere. Federated identity management extends the capability and dynamism of the WebSphere middleware platform with support for federated business interactions.

- ▶ The addition of federated identity management capability can extend the reach of the middleware platform. Services deployed on the WebSphere platform can now be extended to a number of third-party clients and their users.
- ▶ Federated identity management enables WebSphere platform users to access various third-party services with simplified single sign-on. This requires no changes to existing Web applications or Web sites.
- ▶ Tivoli Federated Identity Manager can add significant value to WebSphere Portal by securely connecting portal users with various third-party and software-as-services providers. By delivering Liberty, WS-Federation, and SAML identity *dialects*, FIM helps organizations use the Portal to manage *customer-for-life* scenarios by enabling the portal to transparently bring in third-party resources and delivering these services to portal users without any changes in user experience.
- ▶ Tivoli Federated Identity Manager can add significant value to the WebSphere Business Integration platform. The ability to broker multiple forms of identity enables WebSphere business integration services to implement mediation services connecting various users to various services.

The Tivoli Federated Identity Manager solution itself relates very closely to WebSphere in the sense that Tivoli Federated Identity Manager is an application based on the J2EE specification, and runs on the WebSphere Application Server.

5.5.1 Integrated Solutions Console (ISC)

The IBM Integrated Solutions Console 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 uses ISC to manage and configure FIM domains, federation partners, Web services security partners, keystores, and the trust service.

Archived

Archived



Part 2

Customer environment

Part 2 discusses how identity federation might be used in customer situations.

A scenario that involves several hypothetical corporations is introduced, and it shows how they might be able to take advantage of identity federation to improve customer experiences, reduce costs, and improve overall security.

This scenario, involves two large corporations with internal employee portals. The employees of these corporations authenticate to their corporate portals and are offered access to the service provided by other companies without having to re-authenticate.

Archived

Overview

In this chapter we introduce several hypothetical corporations and show how they might be able to take advantage of identity federation to improve customer experiences and reduce cost and improve overall security.

These use cases and their configurations with Tivoli Federated Identity Manager are covered in detail in subsequent chapters.

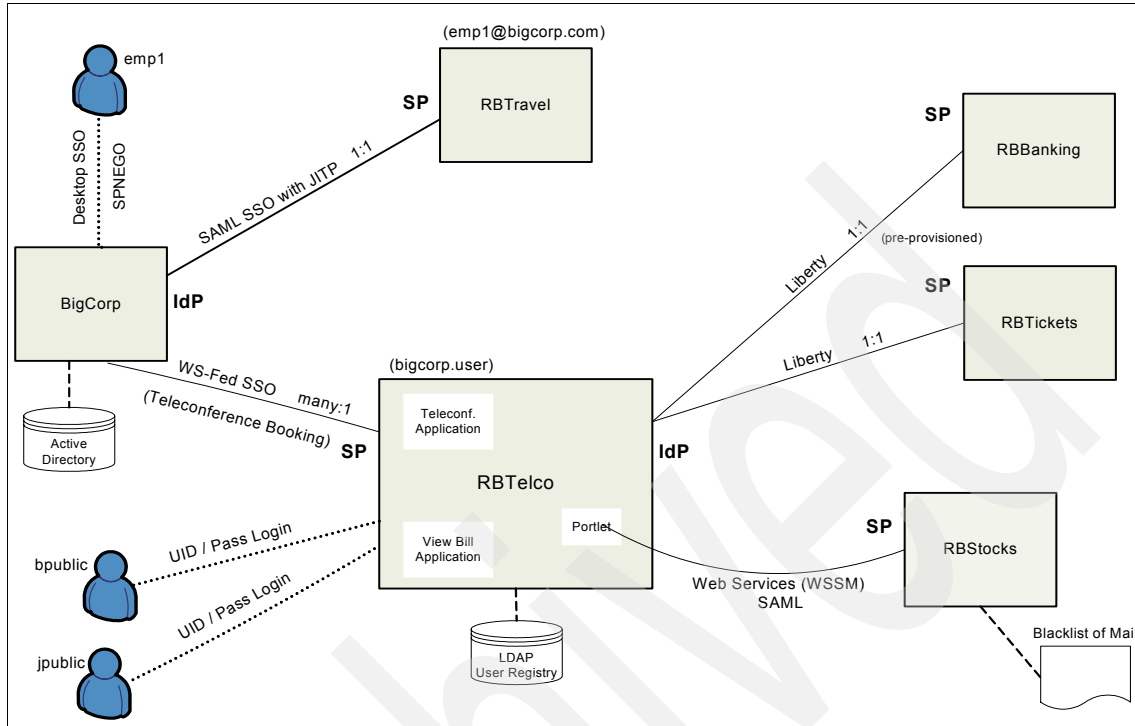


Figure 6-1 Overall scenario logical architecture

The corporations involved in our scenario are:

- ▶ BigCorp - A large organization with a pool of employees using Windows and Linux workstations. They make use of Microsoft's Active Directory for a corporate user registry and have implemented integrated sign-on from Windows workstations to their internal employee portal using SPNEGO¹.
- ▶ RBTelco - A large telecommunication company servicing both individuals (retail customers) and corporate customers. In an effort to provide a rich user experience, RBTelco has partnered with its corporate customers and service providers (RBBanking, RBTickets, and RBStocks) to leverage single sign-on and federation technologies. Through effective application of these technologies, RBTelco is able to deliver seamless interactions for its customers using browsers and mobile devices.
- ▶ RBTravel - A service provider company offering travel booking services for corporate customers. RBTravel maintains user profiles for individuals, but does not support direct authentication or retail customers.

¹ SPNEGO (Simple and Protected GSS-API Negotiation Mechanism) is the subject of rfc2478 and is a way for communicating systems to choose a security mechanism to use.

- ▶ RBBanking - A progressive retail banking corporation looking for new ways to serve its customers. RBBanking has partnered with RBTelco to provide a new service to users who are both RBTelco retail customers and RBBanking customers. These customers are able to link to their account information pages at RBBanking from RBTelco's customer portal. They can check their account information at RBBanking without having to log in to RBBanking.
- ▶ RBTickets - A ticket company selling tickets on the Internet, and is looking for new ways to serve its customers even better. RBTickets has partnered with RBTelco to provide a new service to users who are current RBTelco retail customers. These customers are able to link to and buy tickets from RBTickets through the RBTelco's customer portal. They can do the transaction at RBTickets without having to log in to RBTickets.
- ▶ RBStocks - A company providing a Web services interface to obtain stock quotes. RBStocks accepts requests from RBTelco for two different classes of customers. RBStocks provides real-time stock quotes to RBTelco's corporate customers, and delayed stock quotes to RBTelco's retail customers. RBStocks also maintains a blacklist of users (identified by e-mail address) at its discretion for whom it will not issue stock quotes. For example, RBStocks may periodically receive a list of mail addresses from Traders Anonymous, which it will add to its blacklist.

Figure 6-1, "Overall scenario logical architecture" on page 184, shows the overall configuration and lists the protocols used to establish the single sign-on or account federation. It also describes in which role each of the corporations are, that is, the identity provider or service provider. Each of the companies is presented as logical components; the detailed configuration can be found in the following chapters describing the technical aspects of the federation-related communications between the companies.

The interaction diagrams below are logical representations of the use cases that will be considered. In each case, the technical details are explained in the chapters that follow by considering the messages sent from and to each of the corporations involved.

Simple skeleton applications were generated using Rational® Application Developer to allow us to walk through the scenario. The applications have a minimum of context and function, and serve only to allow the users to follow the flow. Where relevant, the application screens show attribute data about the user that was passed between the identity provider and service provider during federated transactions.

6.1 Use case 1 - SAML/JITP

Employee One, an employee at BigCorp, decides to book business travel. He clicks a link from the BigCorp portal (uses Windows or Linux client, SPNEGO, Kerberos, SAML 1.0). When logging into RBTravel, an account for Employee One is automatically provisioned if necessary. The interaction is shown in Figure 6-2.

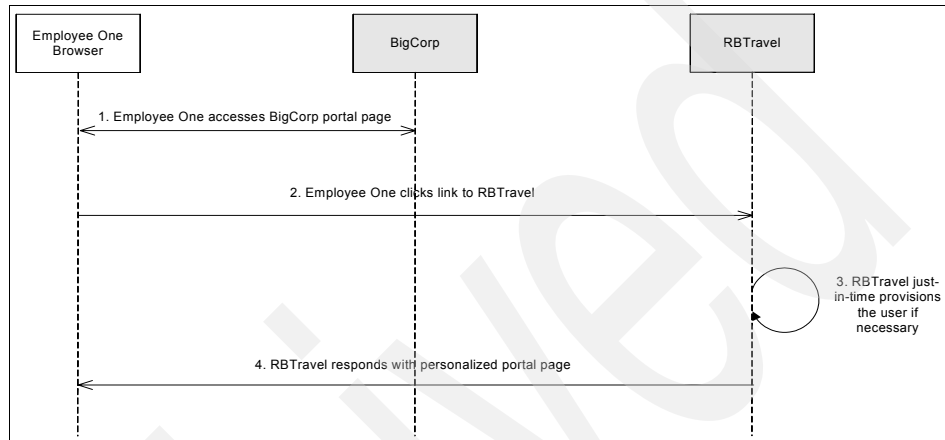


Figure 6-2 Use case 1 high-level interaction

Figure 6-2 is explained below:

1. Employee One accesses the BigCorp employee portal and is automatically authenticated from his desktop workstation. An implicit SPNEGO exchange occurs between John's browser and the employee portal to authenticate John to the portal. Having implicitly authenticated Employee One, the employee portal home page is returned to John's browser.
2. Employee One clicks a link to the RBTravel application.
3. RBTravel, as part of a single sign-on operation, determines whether a local user account and travel profile exists for Employee One. If not, it creates one in real time and authenticates Employee One based on a SAML Assertion exchanged in the single sign-on.
4. RBTravel returns a customized portal application to Employee One's browser.

6.2 Use case 2 - WS-Federation

Employee One, an employee at BigCorp, decides to book a telephone conference for an upcoming business meeting. He clicks a link from the BigCorp portal (using WS-Federation with a many:1 user mapping) and is automatically

authenticated to RBTelco. RBTelco maintains only one user account shared for all BigCorp users; however, during the single sign-on a session credential is created that includes personalization and audit information about the user (display name, e-mail address, and so on) in extended attributes. The interaction is shown in Figure 6-3.

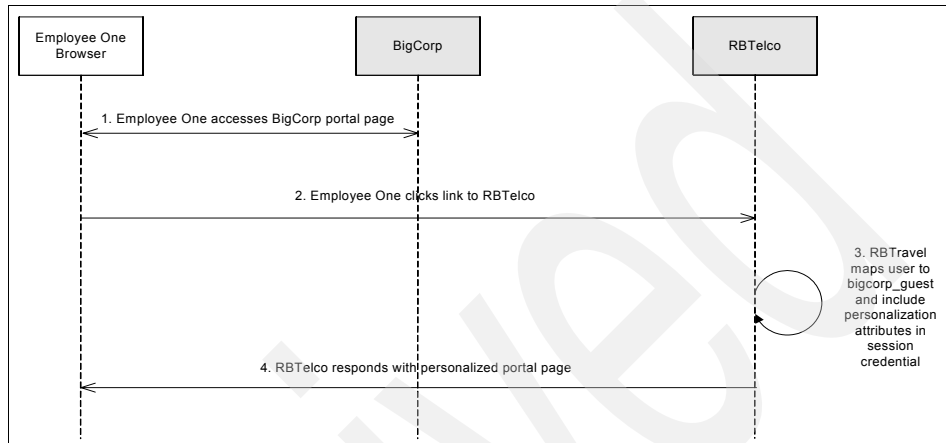


Figure 6-3 Use case 2 high-level interaction

6.3 Use case 3 - Liberty

John Public, a retail customer of RBTelco, also has user accounts at RBBanking and RBTickets. Through a business and technical arrangement between RBTelco and RBBanking, liberty alias data has already been established. There are two main use case flows we discuss:

- ▶ Federating accounts between RBTelco and RBTickets. The high-level user interaction for this flow is shown in Figure 6-4 on page 188.
- ▶ Single sign-in from RBTelco to both RBBanking and RBTickets, followed by single logout. The high-level user interaction for this flow is shown in Figure 6-5 on page 189.

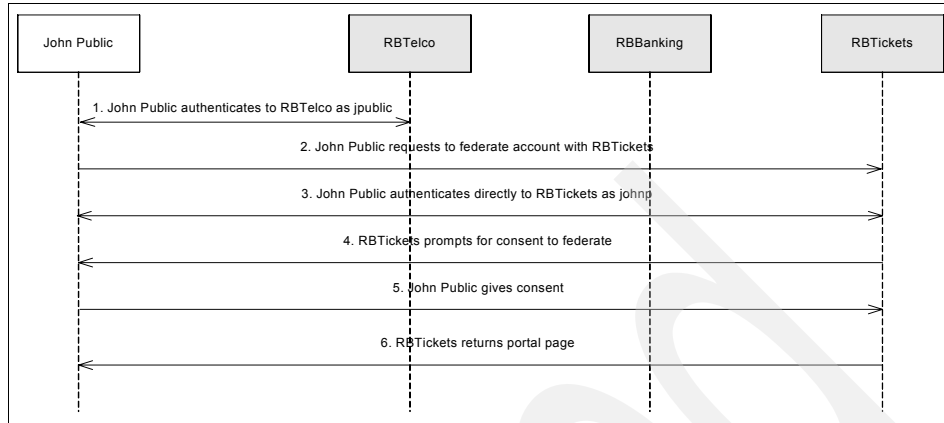


Figure 6-4 Use case 3 high-level interaction for federating accounts

Figure 6-4 is explained below:

1. John Public, a retail customer of RBTelco, authenticates to RBTelco using the user name and password as the user ID jpublic.
2. John Public sees that RBTelco is offering the ability to federate accounts with RBTickets, and he has an account there. He selects the link to federate with RBTickets.
3. Not having yet logged in to RBTickets, authentication is required. John Public logs in to RBTickets with user name and password using his user ID johnp.
4. RBTickets prompts John to ensure that he agrees to federating the accounts. This can be turned off by configuration.
5. John gives consent, and the accounts are federated.
6. John receives the RBTickets portal page.

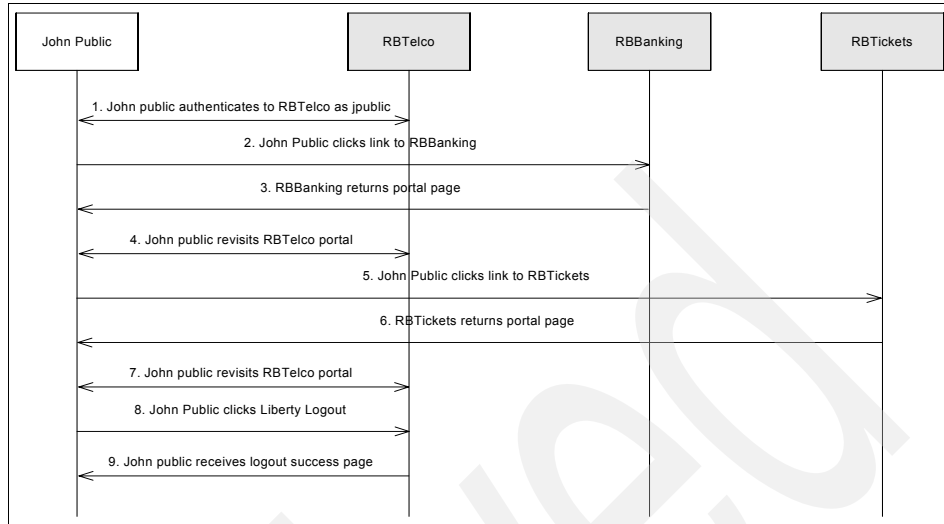


Figure 6-5 Use case 3 high-level interaction for single login and single logout

Figure 6-5 is explained below:

1. John Public, a retail customer of RBTelco, authenticates to RBTelco using user name and password as the user ID jpublic.
2. John selects a link to RBBanking which automatically signs him in as his RBBanking user ID jp.
3. John receives the RBBanking portal application.
4. Later, during the same session, John navigates back to RBTelco.
5. John selects a link to RBTickets. Having previously federated there, he is automatically signed in to RBTickets with his user ID johnp.
6. Later John navigates back to RBTelco. Actually, this is not at all significant, since single logout can be initiated from either of the service providers or the identity provider. Our simple demonstration applications only happen to show a logout link at RBTelco.
7. John clicks logout, and is logged out of RBTelco and all partners to which he has signed in (RBBanking and RBTickets).
8. John receives the logout success page.

6.4 Use case 4 - Web services security management

This use case has two primary types of actors, and three main resulting scenarios. The three scenarios have exactly the same number of flows, as depicted in Figure 6-6.

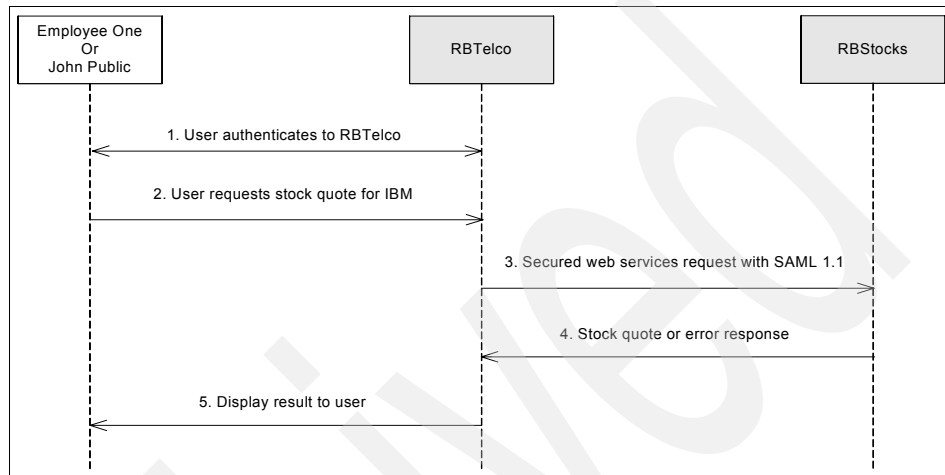


Figure 6-6 Use case 4 high-level interaction

The scenario variants of the use case are described below:

- ▶ Real-time Stock Quote
 - a. Employee One logs into his desktop at BigCorp, and from the BigCorp portal clicks a link to RBTelco, where he is automatically signed on using WS-Federation, as shown in 6.2, “Use case 2 - WS-Federation” on page 186.
 - b. On the RBTelco portal page, there is an option to get a stock quote. Employee One selects the stock quote application, and requests a quote for IBM.
 - c. A secured Web services request is sent from RBTelco to RBStocks containing a SAML 1.1 assertion representing Employee One. The securing of the Web services request is done with a combination of Tivoli Federated Identity Manager Web services security management and WebSphere Web service security technology components. The SAML assertion is built by Tivoli Federated Identity Manager from the Access Manager session credential information at RBTelco, and includes:
 - Employee One’s e-mail address (emp1@bigcorp.com)
 - An attribute indicating this user originated from BigCorp

- d. RBStocks receives and validates the secured Web services request. Tivoli Federated Identity Manager Runtime and Tivoli Federated Identity Manager Web services security management technology is used to validate the SAML assertion and perform identity mapping. As the e-mail address in the assertion is not blacklisted, and since the user came from a business partner of RBTelco (that is, BigCorp), RBStocks responds with a real-time stock quote.
 - e. RBTelco returns an HTML display of the reply for Employee One.
- ▶ Delayed Stock Quote
- a. John Public, a retail customer of RBTelco, logs into RBTelco using his user name and password.
 - b. On the RBTelco portal page, there is an option to get a stock quote. John selects the stock quote application, and requests a quote for IBM.
 - c. A secured Web services request is sent from RBTelco to RBStocks containing a SAML 1.1 assertion representing John. The securing of the Web services request is done with a combination of Tivoli Federated Identity Manager Web services security management and WebSphere Web service security technology components. The SAML assertion is built by Tivoli Federated Identity Manager from the Access Manager session credential information at RBTelco, and includes:
 - John Public's e-mail address (jpublic@rbtelco.com)
 - An attribute indicating this user originated from RBTelco
 - d. RBStocks receives and validates the secured Web services request. Tivoli Federated Identity Manager Runtime and Tivoli Federated Identity Manager Web services security management technology is used to validate the SAML assertion and perform identity mapping. As the e-mail address in the assertion is not blacklisted, and since the user is a retail customer of RBTelco, RBStocks responds with a delayed stock quote.
 - e. RBTelco returns an HTML display of the reply for John.
- ▶ Blacklisted User
- a. RBStocks has a blacklist of e-mail addresses for which it will not issue stock quotes. John Public's e-mail address (it could equally be Employee One) is added to the blacklist. Then John logs into RBTelco using his user name and password.
 - b. On the RBTelco portal page, there is an option to get a stock quote. John selects the stock quote application, and requests a quote for IBM.
 - c. A secured Web services request is sent from RBTelco to RBStocks containing a SAML 1.1 assertion representing John. The securing of the Web services request is done with a combination of Tivoli Federated Identity Manager Web services security management and WebSphere

Web service security technology components. The SAML assertion is built by Tivoli Federated Identity Manager from the Access Manager session credential information at RBTelco, and includes:

- John Public's e-mail address (jpublic@rbtelco.com)
 - An attribute indicating this user originated from RBTelco
- d. RBStocks receives and validates the secured Web services request. Tivoli Federated Identity Manager Runtime and Tivoli Federated Identity Manager Web services security management technology is used to validate the SAML assertion and perform identity mapping. As the e-mail address in the assertion is blacklisted, authorization of the requests fails, and RBStocks responds with an error.
- e. RBTelco returns a HTML error page to the user.

6.5 Conclusions

This concludes the overview of the business use cases considered in this book. The chapters that follow delve into the technical details of how these scenarios were configured and the key implementation and integration tasks performed. Each use case is presented in full in a separate chapter, showing both the identity provider and service provider configurations.

Use case 1 - SAML/JITP

In the following chapter we take a closer look at a very common real-world scenario of federated identity management. Our first use case is a mixture of a simple 1:1 identity mapping and just-in-time-provisioning (JITP) of the identity provided by an identity provider (IdP) to a service provider (SP). The JITP allows the usage of personalized information at the SP without the drawbacks that a normal 1:1 solution have.

To spice things up we have used the Windows integrated Desktop Single-Sign-On (SSO) so that the user experience is like a barrier-free solution. This will raise the end users' acceptance and satisfaction, which is of course a good thing.

7.1 Scenario details

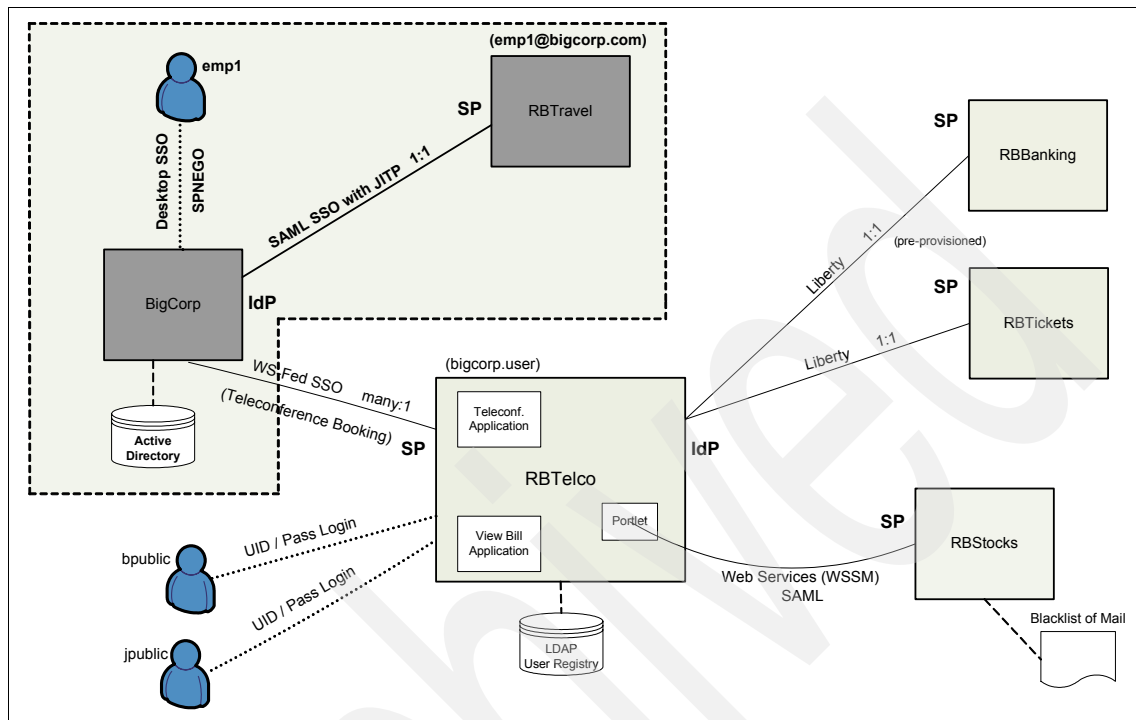


Figure 7-1 Use case 1 logical architecture

We focus on Employee One (emp1) of BigCorp, who logs into his desktop and opens a Web browser with the BigCorps intranet portal. While doing so he automatically gets signed in by the integrated Desktop Single-Sign-On using SPNEGO. Using the portals integrated link to the RBTravel home page he not only gets single signed-on to RBTravel, but his data will be provisioned just in time if they are not already present in the destination system.

The components and actors that are present in this use case are highlighted by the grey box in the upper left corner of the diagram shown in Figure 7-1, “Use case 1 logical architecture” on page 194.

7.1.1 Contract

The very first step in setting up a relation between an IdP and SP is clarifying the technical details of how and what data will be exchanged.

The exchange of intercompany data is a very sensitive issue and will be influenced by many factors before a contract between an identity and a service provider can be signed. We stick with the technical facts to ease the understanding of this part and to have the drivers for the federation configuration.

BigCorp and RBTravel have agreed to federate identities using SAML Version 1.0 using the Browser/Artifact Profile over a HTTPS connection. The SOAP back channel will be using mutually authenticated SSL with a client certificate.

The SAML 1.0 assertion will be signed. The SAML Subject name identifier will contain the e-mail address of the user. There will be another attribute passed in the attribute list of the assertion—the display name of the user for personalization at RBTravel. Example 7-1 shows a sample signed SAML assertion, including its SOAP envelope, confirming the format and name spaces of the attributes.

Example 7-1 Sample SAML assertion passed from BigCorp to RBTravel

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header></soapenv:Header>
  <soapenv:Body>
    <samlp:Response xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      InResponseTo="uuid2adb9caf-0105-fac1-80d7-fae33ca96775"
      IssueInstant="2005-07-18T16:51:40Z" MajorVersion="1" MinorVersion="0"
      ResponseID="FIMRSP_2adb9d94-0105-ec8d-afc1-8ce3efd72411">
      <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"></samlp:StatusCode>
      </samlp:Status>
      <saml:Assertion
        AssertionID="Assertion-uuid2adb9cee-0105-fe74-40da-8ce3efd72411"
        IssueInstant="2005-07-18T16:51:39Z" Issuer="https://www.bigcorp.com"
        MajorVersion="1" MinorVersion="0">
        <saml:Conditions NotBefore="2005-07-18T16:41:39Z"
          NotOnOrAfter="2005-07-18T17:01:39Z">
          <saml:AudienceRestrictionCondition>
            <saml:Audience>https://www.rbtravel.com/ITFIM/sps/samlfed/saml/login</saml:Audience>
          </saml:AudienceRestrictionCondition>
        </saml:Conditions>
        <saml:AuthenticationStatement
          AuthenticationInstant="2005-07-18T16:51:39Z"
          AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
          <saml:Subject>
            <saml:NameIdentifier
              Format="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">emp1@bigcorp.com</saml:NameIdentifier>
          </saml:Subject>
        </saml:AuthenticationStatement>
      </saml:Assertion>
    </samlp:Response>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:artifact-01</saml:Confir
mationMethod>
        </saml:SubjectConfirmation>
    </saml:Subject>
</saml:AuthenticationStatement>
<saml:AttributeStatement>
    <saml:Subject>
        <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">emp1@bigcorp.com</s
aml:NameIdentifier>
        <saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:artifact-01</saml:Confir
mationMethod>
        </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute AttributeName="cn"
AttributeNamespace="http://www.bigcorp.com/cn">
        <saml:AttributeValue>Employee One</saml:AttributeValue>
    </saml:Attribute>
</saml:AttributeStatement>
<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/xmlsig#rsa-sha1"></ds:SignatureMethod>
        <ds:Reference
URI="#Assertion-uuid2adb9cee-0105-fe74-40da-8ce3efd72411">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmlsig#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/xmlsig#sha1"></ds:DigestMethod>
            <ds:DigestValue>XJieAD/CpXPPw3q6wn0u2i0LwsA=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>

```

```

<ds:SignatureValue>BYV5yZ8QY3b8aKm9zaQqmrGIWfYaLwcDUER5sp7Bgn4i2c/SEk2DErT2z0dW
/nZR2i7uhQ1OZDfu2PrB/ruv3kyMJUVyuy2wHD2Ro4SgQ4kYbxyg6GR0tzJC2Cx+EfQz4ai0IbV7eKO
LF+NZ0hBj2kpb/8TobqTzgOK9L803UkE=</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>

<ds:X509Certificate>MIICqjCCAhOgAwIBAgIBAzANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQQDExN
maW0ucmVkJm9vay5pYmOuY29tMQswCQYDVQQGEwJVUzEMMAoGA1UEChMDSUJNMB4XDTA1MDYwMTIxMj
QzOV0XDTEwMDYxNjIxMjQzOVowRjE1MCMGA1UEAxQcYmInY29ycF9yYnRyYXZlbc5iaWdj3JwLmNvb
TElMAkGA1UEBhMCVVMxEADA0BgNVBAoTB0JpZ0NvcnAwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGB
AL+up7hI0vMJB/g9Zhg1KTW3x/PxTVhG516hJ3kNdrZeJhPg59usfWmrEJSn2Ug1EGSGz5kTWvYS2dn
AMANDAoWESTMANgbyzLdp0b2iLKSLeKyRcRk+u6i6Hbs8g0zoLGJyv+zZaOLSUY0j186SrGb8L575PA
Ws5j1kwPIULohPAGMBAAGjgbQwgbEwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUWw6uNP9izCSYZ04Tt
eb6Sa9bx2owYQYDVR0jBFowWIAUQNM+0+Jvv8jfpobQbQhsXg/LkTghPaQ7MDkxHDAaBgNVBAMTE2Zp
bS5yZWRib29rLmliS5jb20xCzAJBgNVBAYTA1VTMqwwCgYDVQQKEwNjQk2CAQEWcwYDVR0PBAQDAgS
MARTINGgWESTMANiBDQFFgNocGgwdQYJKoZIhvcNAQEEBQADgYEAnmviu+bM1gS3iBSK1w/3X/rZL
3GTPOoM1UCcGhzy1mF0xKe+Bm/lIaqG2qdx2uGjKkeOACjecNM93Je9PfYb7XP1p53C7azCOZIsOeiw
fTRDShWtQqQo0wduIYafJSeQNN14zakIxCREVSKUXs2eQdBCLi4KVxHN8Zg6W1xwdQ=</ds:X509Cer
tificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
</saml:Assertion>
</samlp:Response>
</soapenv:Body>
</soapenv:Envelope>

```

7.1.2 User experience

Our user, Employee One, will access three systems in this use case. For accessing all systems, he will authenticate himself only once by logging into his desktop system. This could be done by providing a user and password, but also by any other valid authentication mechanism like token systems, secure cards, biometric services, and so on. The point is that after the user is validated by the system he is able to access all systems without any further visible authentication of the user.

So the user's first step is to log into his desktop, which can be seen in Figure 7-2 on page 198 for Linux or Figure 7-3 on page 198 for Windows. As you see, the user name and password have to be entered once.

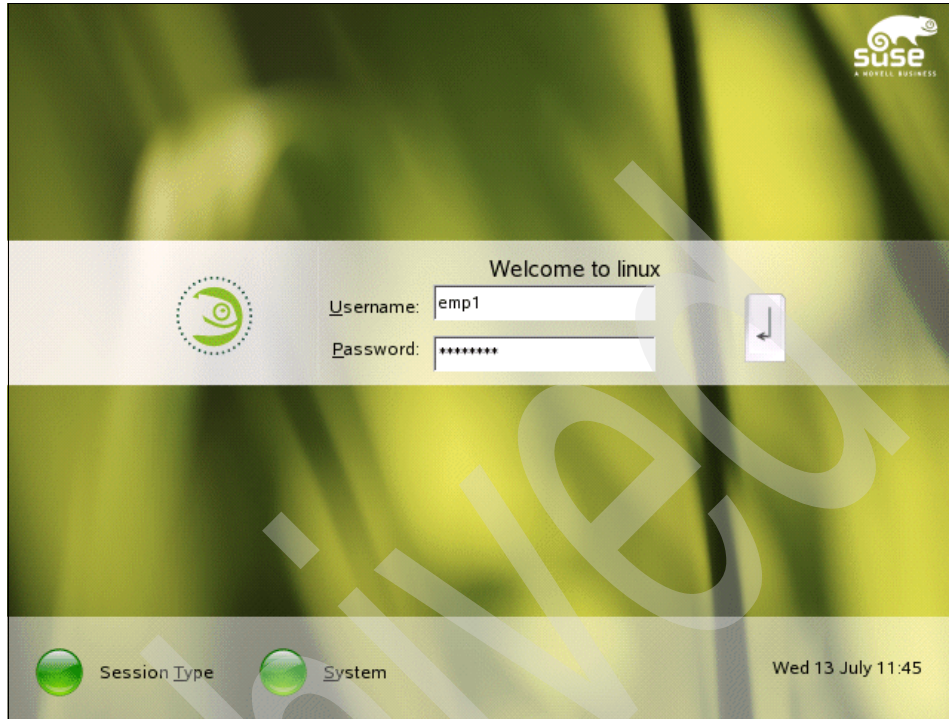


Figure 7-2 Employee One logs on to his Linux desktop



Figure 7-3 Employee One logs on to his Windows desktop

The second step is to open a browser and point to the BigCorps Intranet portal, as in Figure 7-4 on page 199. Please notice that the user name appears in the

title and inside the body of the page. This is possible because the user has been signed on using SPNEGO without any special intervention.

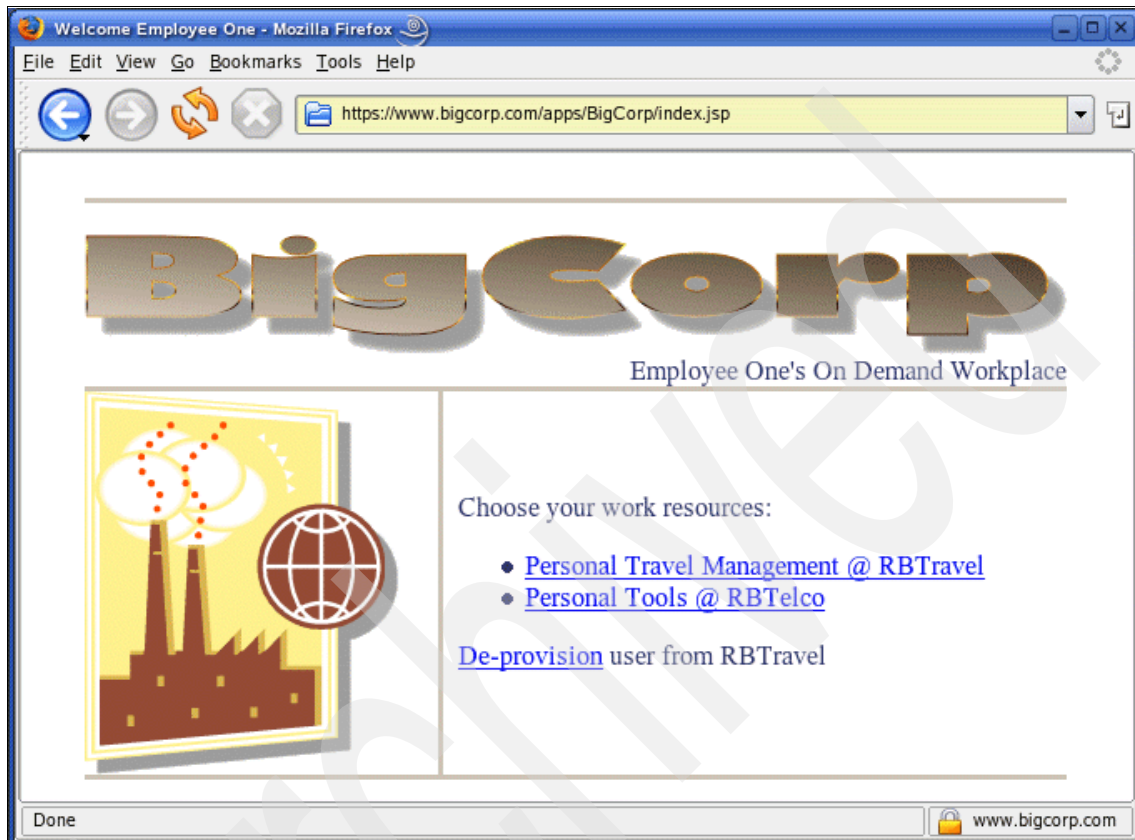


Figure 7-4 BigCorp Portal Intranet page

Most people are used to this automatic logon from using Microsoft's Internet Explorer (IE since Version 5.01) and the Internet Information Server (IIS since Version 5.0). We wanted to show that it works perfectly with other browsers and platforms. For more information visit:

<http://www.mozilla.org/projects/netlib/integrated-auth.html>

Now that our Employee One is already authenticated to the BigCorp Intranet portal, he heads towards RBTravel and clicks the **Personal Travelmanagement @ RBTravel** link.



Figure 7-5 Employee One at RBTravel's site

After the user has selected the link to RBTravel, several invisible things happen, and he will arrive at the screen shown in Figure 7-5.

Please note that the user's full name appears inside the body of the page. Without any further interaction, the user has been signed on to the destination. Even more, the user's data have been provisioned while doing this the first time. This just-in-time-provisioning (JITP) is described in depth in 7.2.2, "Single sign-on - SAML/JITP" on page 201.

You may note that the user has the option to remove his provisioned account data by clicking the **Remove this provisioned BigCorp user** link. In our case it is realized with a simple CGI script that provides this functionality. See the result in Figure 7-6 on page 201.

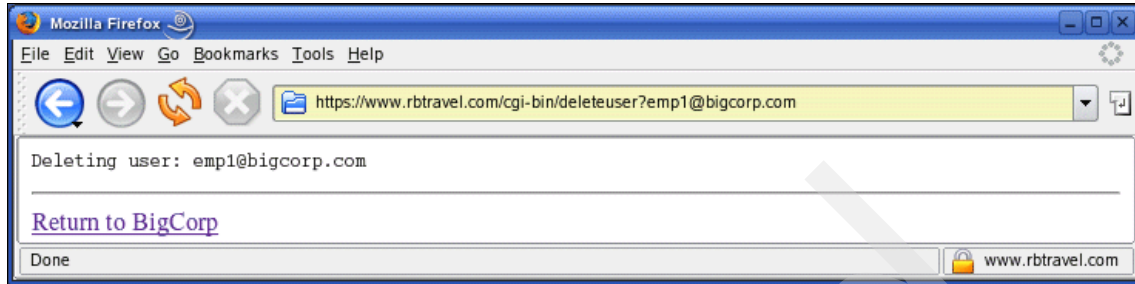


Figure 7-6 User is deleted from the service provider

7.2 Functionality

The provided functionality for this scenario includes an SPNEGO and a SAML/JITP based single sign-on.

7.2.1 Single sign-on - SPNEGO

One functionality that this use case provides is a Desktop SSO using the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO). SPNEGO is a GSS-API (RFC 2478) based protocol that defines a Kerberos logon over HTTP. Thus the user's desktop logon credentials will be used to authenticate him against a Web server.

For more information about using SPNEGO with Access Manager see:

<http://www.ibm.com/developerworks/tivoli/library/t-ss/>

7.2.2 Single sign-on - SAML/JITP

The second functionality we face in this use case is an intercompany SSO based on SAML Version 1.0 as defined by OASIS. For more details about SAML see:

<http://www.oasis-open.org/>

But SAML is only the transport mechanism. The important part of this scenario is processing the user's information from the provided SAML token and polling the local user registry to see if this user already exists in our domain. If yes, they are logged in. If not, we just-in-time provision the user into our local registry, and then log them in. This way no extra synchronization has to be established between BigCorp and RBTelco.

7.3 Partners involved

The corporations involved in this use case are BigCorp and RBTravel.

7.3.1 BigCorp

BigCorp provides an employee portal to access its internal systems. It also provides an access point to services external to the company. These are typically either employee benefits related or services provided by their business partners.

BigCorp has entered into agreements with their business partners to provide these enhanced services through their corporate portal. These agreements include the technical “creation” of the federation relationship between the two parties.

7.3.2 RBTravel

RBTravel provides all necessary services for a successful and pleasant journey. They are specialized to offer these services to companies as an integrated service to their portals.

BigCorp is one of RBTravel's customers that gets personalized services for each user of BigCorp.

7.4 Interaction description

This use case description has been split in two parts: First, where the BigCorp Employee One authenticates with his workstation and then accesses his company portal. Secondly, he is heading from the portal over to his travel information provider RBTravel.

7.4.1 High-level Interaction overview

Before we deep dive into every step of the interaction we will have a look at the basic steps that the user is taking, as described in 7.1.2, “User experience” on page 197.

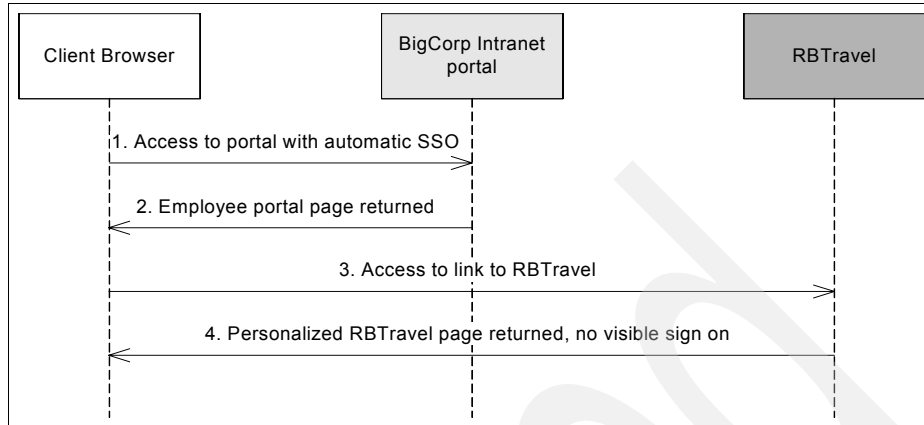


Figure 7-7 High-level Interaction diagram

The steps shown are:

1. Employee One accesses the BigCorp one employee portal and is automatically authenticated from his workstation. An implicit SPNEGO exchange occurs between the browser and the portal to authenticate the user.
2. Selected user information has been transported from BigCorps WebSEAL as headers to the real portal so that this information has been taken to create a personalized page for the user.
3. The user clicks a link in the portal that will bring him to the RBTravel Web site. By doing so he gets provisioned just in time if necessary.
4. Similar to the second step, the user receives a personalized page back.

7.4.2 Single sign-on from Windows workstation (SPNEGO)

As described in 7.2.1, “Single sign-on - SPNEGO” on page 201, a user will have a desktop SSO with a Web server. This functionality is just a piece in the chain of a barrier free user experience, but it does not belong to this book’s theme about Tivoli Federated Identity Management. Hence, we only give a very rough overview of the involved interactions from step 1 and 2 of Figure 7-7:

1. The steps are:
 - a. The user logs into the desktop using a user and password. Note that in case the desktop would allow other login forms like ID cards, fingerprint reader, face recognition, and so on they would all be valid login methods.
 - b. The user opens a browser with the BigCorp portal URL.
 - c. WebSEAL requests authorization using SPNEGO.

- d. The browser answers the authorization request.
 - e. WebSEAL forwards the request to the backend server.
2. The steps are:
- a. The backend server sends answer.
 - b. WebSEAL forwards the answer to the browser.

7.4.3 Single sign-on from BigCorp to RBTravel (SAML/JITP)

After the user has been SSOed to the desktop and intranet, he will now face the BigCorp portal. The next step is to advance over to the RBTravel site. He will do so by clicking a prepared URL that will take care of the site transfer.

We also depict the just-in-time-provisioning (JITP) that the user is not able to see because the whole process will take place in the background.

Note: Even though the process flow in this use case starts at the IdP (BigCorp) portal site this does not mean it could not be started at the SP (RBTravel) site. In fact, the intersite transfer could be launched perfectly at the SP site also.

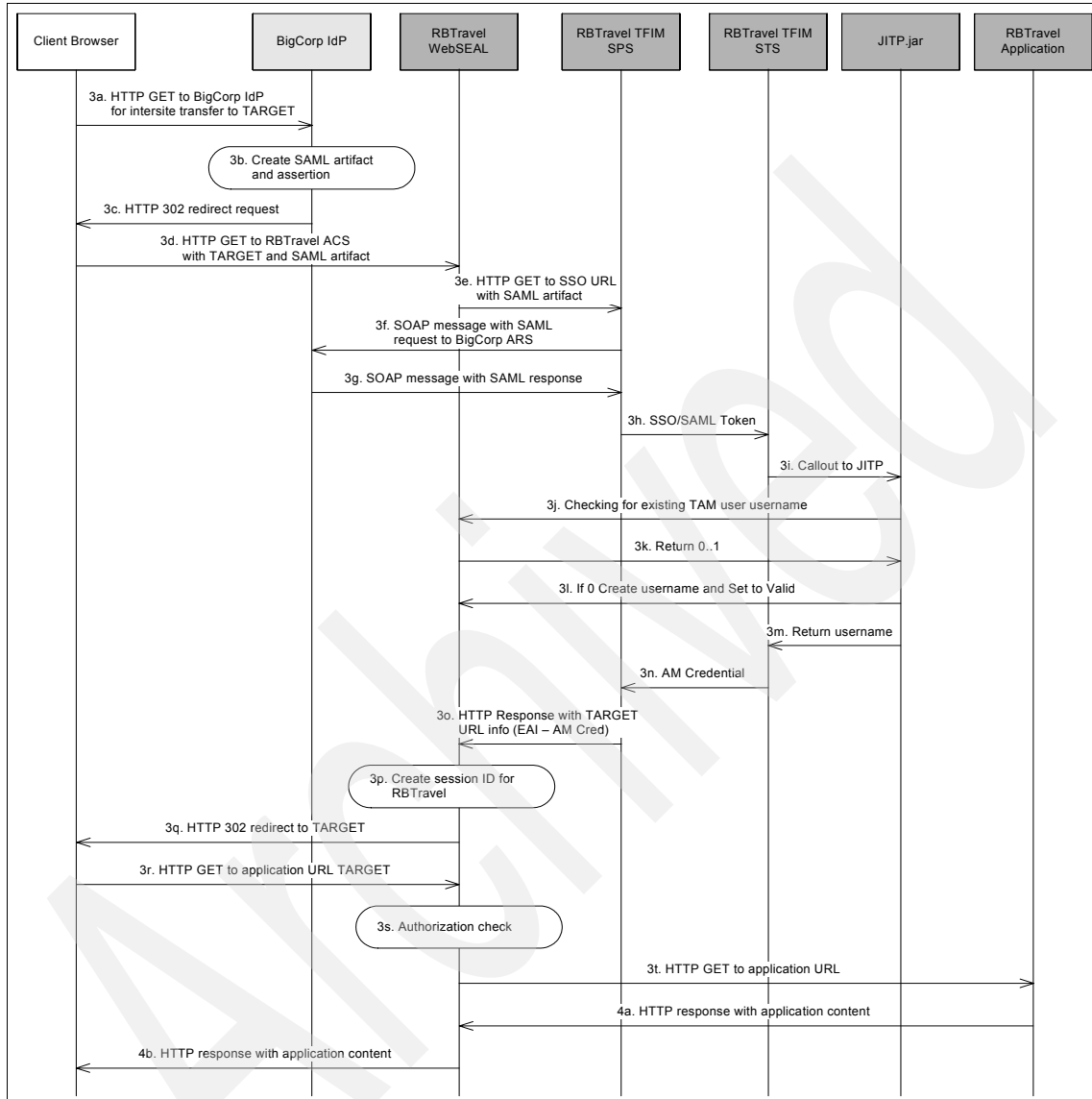


Figure 7-8 SAML Browser/Artifact Profile flow with JITP

Remember that the user has already signed in to his desktop and intranet, as described in the previous chapter. Now we express steps 3 and 4, shown in Figure 7-7 on page 203, of the use case in more detail:

3. The steps are:

- a. By clicking an arranged URL that initiates the intersite-transfer-service, the user starts the SSO from the IdP to the SP:

```
https://www.bigcorp.com/ITFIM/sps/samlfed/saml/login?TARGET=https://www.rbrtravel.com/apps/RBTravel/index.jsp
```

- b. As we already have a session with the IdP, the SSO Protocol Service (SPS) at BigCorp now creates the assertion and its corresponding artifact. The assertion will be stored in memory and the artifact will be attached to the URL that the browser receives back, like this example:

```
https://www.rbrtravel.com/ITFIM/sps/samlfed/saml/login?SAMLart=AAF1sv0siM1dXMY2%2BjJhh5NZzf1YmdoWiS%2BuuTsAFEB4%2BvxIagsd%2Btk&TARGET=https://www.rbrtravel.com/apps/RBTravel/index.jsp
```

The artifact is constructed of a version number (TypeCode), the provider's succinct ID (SourceID), and a random number (AssertionHandle). The base64 encoded artifact is used as a pointer to its corresponding assertion.

- c. The user's browser receives the above created URL with a HTTP 302 redirect request.
- d. Now the browser automatically uses the redirect to the destined URL.
- e. The SP WebSEAL forwards the request to RBTravels SPS.
- f. Invisible to the user, the SPs Assertion Consumer Service (ACS) sends a SAML SOAP message to the IdPs Assertion Resolution Service (ARS) as a HTTP POST.
- g. The ARS at the IdP now sends the SAML assertion back to the ACS.
- h. The SPS now passes over the SSO/SAML token to the Secure Token Service (STS) inside the trust service and validates the incoming request.
- i. At this point we leave the normal SAML Browser/Artifact Profile path for a few more stops to introduce the JITP. Instead of just mapping the information pieces from inside the SAML token, we call out for a little Java JAR that is referenced inside the XSL mapping (see "BigCorp mapping for use case 1" on page 404).
- j. The JAR tries to find an already existing Access Manager user over API calls.
- k. Either no user or one user can be located.

- l. In case no user was found, the JAR now creates and activates a user over the API. By doing so the user will be provisioned the first time he ever signs on to the SP without any intervention of external interfaces or synchronization.
 - m. The last step of our little Java magic is to send back the Access Manager user name to the STS and fulfill the required user mapping.
 - n. With the delivered Access Manager user name, the STS is now able to build a valid Access Manager credential and issues this information back to the SPS.
 - o. After the user now has been SSOed RBTravels SPS creates a HTTP response to the real TARGET and sends back the credentials using the External Authentication Interface (see “External Authentication Interface” on page 370).
 - p. RBTravels WebSEAL intercepts the incoming response and creates a user session.
 - q. The next step is again visible to the user because RBTravels WebSEAL is sending a HTTP 302 redirect request to the browser with the TARGET URL as location.
 - r. The browser picks up the redirect request and finally loads the TARGET URL as requested with the first step.
 - s. RBTravels WebSEALs makes an authorization check and allows access to the protected destination.
 - t. The WebSEAL now sends the request to the junctioned application.
4. Step 4 is:
- a. A HTTP response is delivered back from the application server to RBTravels WebSEAL.
 - b. Finally the requested content is delivered to the user’s browser.

For more information about the Java Code and how to use it inside the XSLT mapping see “Calling Java code from mapping rules” on page 399.

7.5 Configuration data

The following chapters describe the configuration for the federation between BigCorp and RBTelco.

The assumption is that Tivoli Federated Identity Management is already installed and the runtime deployed and configured.

The following references assist with the installation and configuration of Tivoli Federated Identity Management:

- ▶ *IBM Tivoli Identity Manager Installation Guide Version 6.0*, GC32-1667-00, discusses the installation of Tivoli Federated Identity Management.
- ▶ *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00, contains basic information about configuring the Tivoli Federated Identity Management runtime, and information on configuring federations.
- ▶ Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, contains information about configuring Tivoli Federated Identity Management for use with WebSEAL

7.5.1 IdP-related configuration data

First we show the federation configuration data, and after this we have the partner configuration data.

Configuring a SAML Federation at BigCorp consists of the following tasks:

- ▶ Importing BigCorp signing keys
- ▶ Configuring Tivoli Federated Identity Management using SAML as an identity provider
- ▶ Configuring a service provider partner for RBTravel
- ▶ Configuring an Access Manager policy for the federation URLs

Importing BigCorp keys

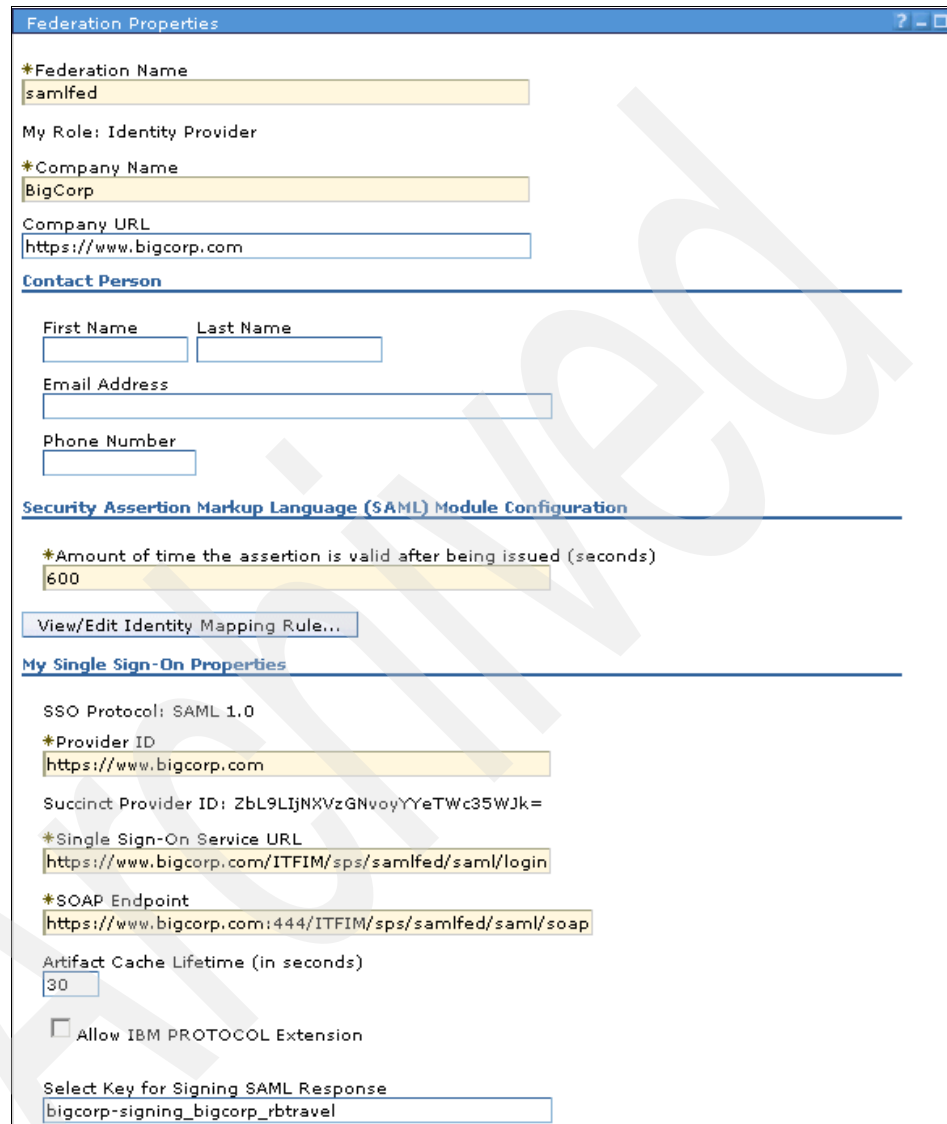
Appendix C, “Keys and certificates” on page 425, contains information on the key strategy used for all use cases. In particular, note that for this federation configuration the `bigcorp-signing.jks` key file was imported into Tivoli Federated Identity Management. This contains the signing key used to sign the SAML assertion sent to RBTravel.

Note: The format to address a key in a key file is “<keyfile>_<key>”. So in Figure 7-9 on page 209, “bigcorp-signing” is the above-mentioned JKS keystore file and the referenced key is called “bigcorp_rbtravel”, which assembles to “bigcorp-signing_bigcorp_rbtravel”.

Configuring BigCorp as a SAML identity provider

Detailed information on configuring an identity provider for using SAML is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*,

GC32-1668-00. This section discusses the specific configuration parameters used for BigCorp.



Federation Properties

*Federation Name
samlfed

My Role: Identity Provider

*Company Name
BigCorp

Company URL
https://www.bigcorp.com

Contact Person

First Name Last Name

Email Address

Phone Number

Security Assertion Markup Language (SAML) Module Configuration

*Amount of time the assertion is valid after being issued (seconds)
600

[View/Edit Identity Mapping Rule...](#)

My Single Sign-On Properties

SSO Protocol: SAML 1.0

*Provider ID
https://www.bigcorp.com

Succinct Provider ID: ZbL9LIjNXVzGNvovYYeTWc35WJk=

*Single Sign-On Service URL
https://www.bigcorp.com/ITFIM/sps/samlfed/saml/login

*SOAP Endpoint
https://www.bigcorp.com:444/ITFIM/sps/samlfed/saml/soap

Artifact Cache Lifetime (in seconds)
30

Allow IBM PROTOCOL Extension

Select Key for Signing SAML Response
bigcorp-signing_bigcorp_rbtravel

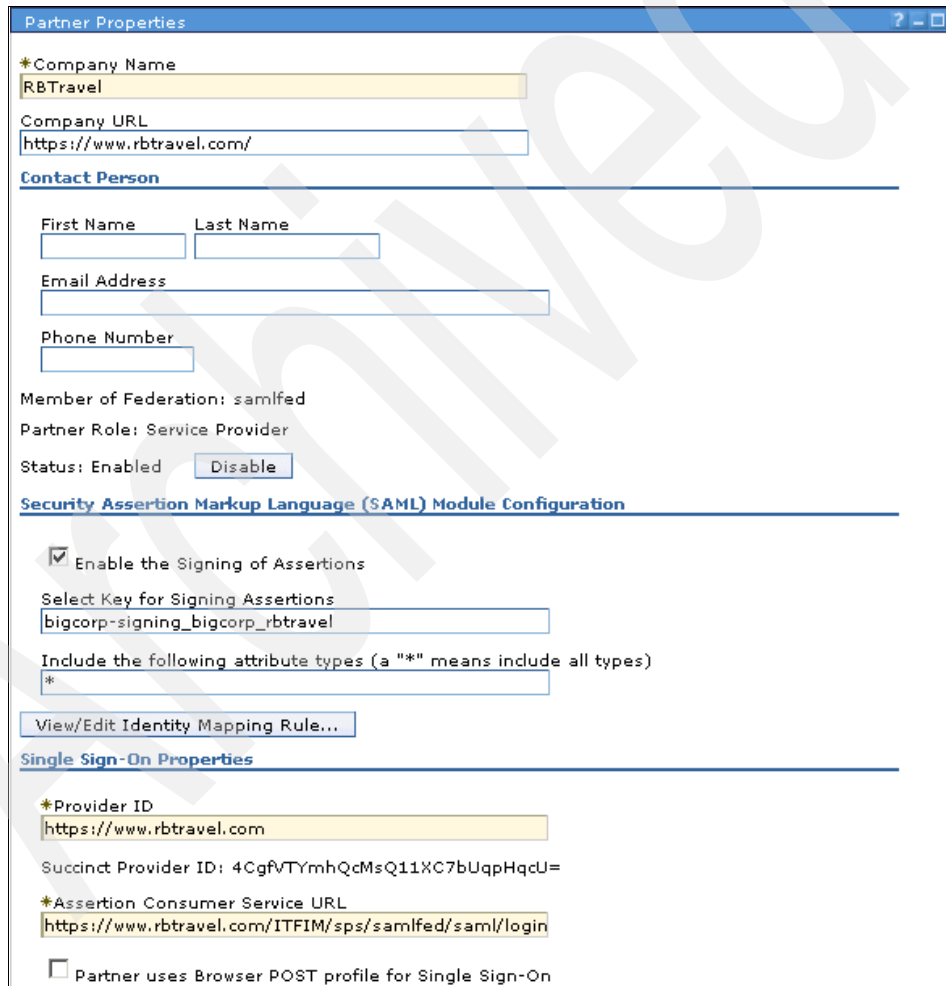
Figure 7-9 IdP SAML federation configuration summary page

Figure 7-9 shows all information needed to configure the SAML federation for BigCorp. For more information about the identity mapping including the complete XSLT mapping see “BigCorp mapping for use case 2” on page 410.

Attention: A common mistake for any configuration is to enter the Provider ID with a trailing slash like `https://www.bigcorp.com/`, which will lead to an error when using the configuration. Remember that this is just a configuration parameter and not a link. Future versions of the console may take care of this.

Configuring a service provider partner for RBTravel

Figure 7-10 shows all of the information needed to configure the SAML federation partner RBTravel with BigCorp. The identity mapping rule is already defined in the federation and is therefore left empty in the partner configuration.



Partner Properties

*Company Name
RBTravel

Company URL
https://www.rbtravel.com/

Contact Person

First Name Last Name
[] []

Email Address
[]

Phone Number
[]

Member of Federation: samlfed
Partner Role: Service Provider
Status: Enabled

Security Assertion Markup Language (SAML) Module Configuration

Enable the Signing of Assertions
Select Key for Signing Assertions
bigcorp-signing_bigcorp_rbtravel

Include the following attribute types (a "*" means include all types)
*

Single Sign-On Properties

*Provider ID
https://www.rbtravel.com

Succinct Provider ID: 4CgfVTYmhQcMsQ11XC7bUqpHqcU=

*Assertion Consumer Service URL
https://www.rbtravel.com/ITFIM/sps/samlfed/saml/login

Partner uses Browser POST profile for Single Sign-On

Figure 7-10 IdP SAML federation partner configuration summary

Configuring Access Manager policy at BigCorp

In this case there is actually no specific requirement for an Access Manager configuration of the SAML URL at BigCorp. This is because BigCorp's policy is to allow any employee to access the RBTravel-hosted traveling arrangement application. That being the case, the default-webseal ACL, which allows access to any authenticated user, is suitable. If only some employees were allowed to access the application (for example, frequent travelers), then the following object in BigCorp should be appropriately protected with an Access Manager authorization policy:

```
/WebSEAL/www.bigcorp.com-default/ITFIM/sps/samlfed/saml/login
```

As we are using the SAML Browser/Artifact profile, we have to set up some protective measures to deny the access to the Tivoli Federated Identity Management SOAP endpoint at the default WebSEAL and only allow access to a special group of users to the SOAP endpoint at the SOAP WebSEAL at port 444. We introduced two ACLs that will take care of this.

The ACL `not_allowed_acl`, as shown in Example 7-2, denies access to objects for normal unauthenticated and authenticated users and is attached to the following two objects:

```
/WebSEAL/www.bigcorp.com-default/ITFIM/sps/samlfed/saml/soap  
/WebSEAL/www.bigcorp.com-soap
```

The first object is the Tivoli Federated Identity Management SOAP endpoint at the default WebSEAL and the second object is the complete SOAP WebSEAL.

Example 7-2 Access Manager not_allowed_acl ACL

```
ACL Name: not_allowed_acl  
Description: Deny access to objects for normal unauth and auth user  
Entries:  
  User sec_master TcmdbsvaBR1  
  Group webseal-servers Tgmdbsrx1  
  Group iv-admin TcmdbsvaBRrx1  
  Any-other T  
  Unauthenticated T
```

As we denied access to the whole SOAP WebSEAL with the above measures, we introduced the ACL `soap_clients_acl`, as shown in Example 7-3 on page 212, which allows access for the group `soap_clients_grp`. This ACL is attached to the SOAP endpoint of the SOAP WebSEAL:

```
/WebSEAL/www.bigcorp.com-soap/ITFIM/sps/samlfed/saml/soap
```

Example 7-3 Access Manager soap_clients_acl ACL

```
ACL Name: soap_clients_acl
Description: Allow access to objects for group soap_clients
Entries:
  User sec_master TcmdbsvaBR1
  Group iv-admin TcmdbsvaBRrx1
  Group webseal-servers Tgmdbsrx1
  Group soap_clients_grp Tr
  Any-other T
  Unauthenticated T
```

The WebSEAL instance is configured to only accept client certificates as the login method. To simplify the mapping of the certificate to the actual user, we have used a very simple CDAS module, shown in Example 7-4 on page 215. This CDAS module is mapping every certificate to the user soapclient, which is of course a member of the soap_clients_grp, and so it is granted access to the SOAP endpoint. A real-world scenario would either use WebSEAL's built-in mapping or a more complex CDAS module.

7.5.2 SP-related configuration data at RBTravel

Configuring SAML at RBTravel consists of the following tasks:

- ▶ Importing RBTravel partner keys
- ▶ Configuring Tivoli Federated Identity Management for SAML as a service provider
- ▶ Configuring an identity provider partner for BigCorp
- ▶ Configuring Access Manager policy for the federation URLs

Importing RBTravel keys

Appendix C, “Keys and certificates” on page 425, contains information about the key strategy used for all use cases. In particular, note that for this federation configuration the rbtravel-partners.jks, and rbtravel-partners.jks key files have been imported into Tivoli Federated Identity Management. These contain the public certificate used to verify the signature on the SAML assertion sent from BigCorp and the client certificate to authenticate against the BigCorp SOAP WebSEAL.

Configuring RBTravel as a SAML service provider

Detailed information on configuring a SAML service provider is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00. This section discusses the specific configuration parameters used for RBTravel.

Federation Properties

*Federation Name
samlfed

My Role: Service Provider

*Company Name
RBTravel

Company URL
https://www.rbtravel.com

Contact Person

First Name	Last Name
John	Admin

Email Address
jadmin@rbtravel.com

Phone Number
555-5555

View/Edit Identity Mapping Rule...

My Single Sign-On Properties

SSO Protocol: SAML 1.0

*Provider ID
https://www.rbtravel.com

Succinct Provider ID: 4CgfVTYmhQcMsQ11XC7bUqpHqcU=

*Assertion Consumer Service URL
https://www.rbtravel.com/ITFIM/sps/samlfed/saml/login

Figure 7-11 SP SAML federation configuration summary

Figure 7-11 shows all of the information needed to configure the SAML federation for RBTravel. For more information about the identity mapping, including the complete XSLT mapping, see “RBTravel mapping for use case 1” on page 405.

Configuring an identity provider partner for BigCorp



Partner Properties

*Company Name
BigCorp

Company URL

Contact Person

First Name Last Name

Email Address

Phone Number

Member of Federation: samlfed
Partner Role: Identity Provider
Status: Enabled

Security Assertion Markup Language (SAML) Module Configuration

Enable Signature Validation

*Select Validation Key
rbtravel-partners_bigcorp_rbtravel

Single Sign-On Properties

*Provider ID
https://www.bigcorp.com

Succinct Provider ID: ZbL9LjNXVzGNvoyYYeTWc35WJk=

*Single Sign-On Service URL
https://www.bigcorp.com/ITFIM/sps/samlfed/saml/login

*SOAP Endpoint
https://www.bigcorp.com:444/ITFIM/sps/samlfed/saml/soap

Select Key for Validating Your Partner's SAML Response
rbtravel-partners_bigcorp_rbtravel

SOAP SSL Connection Parameters (used ONLY if SOAP Endpoint is https)

SOAP Server Authentication Key Identifier
rbtravel-ca_redbook_ca

Use Client Certificate for SOAP

SOAP Client Authentication Key Identifier
rbtravel-clients_rbtravel_bigcorp

Figure 7-12 SP SAML federation partner configuration summary

Figure 7-12 shows all of the information needed to configure the SAML federation partner RBTravel with BigCorp. The identity mapping rule is already defined in the federation and therefore left empty in the partner configuration.

Configuring Access Manager policy at RBTravel

The SAML endpoint at RBTravel is being used to authenticate users to the RBTravel WebSEAL. Consequently, it is necessary to allow unauthenticated access to this URL. “Access Manager policy for trigger URLs for EAI” on page 376 discusses the need for this, and in our case the federation URL to which the unauthenticated-allowed ACL needs to be applied is:

```
/WebSEAL/<webseal_server>/ITFIM/sps/samlfed/saml/login
```

7.6 Assumptions/implementation notes

Though the JITP is very nice idea, it is yet not optimized due to the fact that it calls out to check for the users with every logon. Plans are to introduce a better way of doing so within Tivoli Federated Identity Management.

The following part shows the CDAS source code that has been used to map client certificates to the soapclient user, as described in “Configuring Access Manager policy at BigCorp” on page 211.

Example 7-4 Simple CDAS module mapping certificates to user soapclient

```
static char sccsid[]="@(#)94 1.10 src/ivauthn/modules/pdxauthn/pdxauthn_adk/xauthn.c,  
pdweb.authn, pdweb390, 020409a 3/26/02 16:26:07";  
/*  
 * FILE: xauthn.c  
 *  
 * PD cross domain authentication (CDAS) demo. This file implements  
 *  
 * xauthn_initialize()  
 * xauthn_shutdown()  
 * xauthn_authenticate()  
 * xauthn_change_password()  
 *  
 * functions used by the PD WebSEAL to authenticate  
 * users based on the specified authentication  
 * mechanism.  
 *  
 * To configure, modify the iv.conf file, under the  
 * [authentication-mechanisms] stanza, select the desired  
 * authentication mechanism that you want this library to  
 * be used, and assign this library to it.  
 *
```

```

* For example, if you wish this library to process all the
* HTTP SSL username/password LDAP authentication, specifies
* the following:
*
*   passwd-ldap = libxauthn.so
*
* or
*
*   passwd-ldap = libxauthn.so<args>
*
* if you have any particular arguments that you want to pass
* to this library for initialization and shutdown.
*
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ogauthzn.h>
#if !defined(WIN32) && !defined(_WIN32)
    #include <strings.h>
#endif

#include "pdxauthn.h"
#include "xattr.h"
#include "xnvlst.h"

/*
* FUNCTION NAME
*   xauthn_initialize
*
* DESCRIPTION
*   init the authentication service
*
* ARGUMENTS
*   [in] argc The count of arguments to the service.
*   [in] argv The array of argument strings.
*
* RETURN VALUE
*   XAUTHN_S_COMPLETE on success, error code on failure
*/
xauthn_status_t
xauthn_initialize(
    int          argc,      /* in */
    const char **argv      /* in */
)
{
    return XAUTHN_S_COMPLETE;
}

```

```

}

/*
 * FUNCTION NAME
 *   xauthn_shutdown
 *
 * DESCRIPTION
 *   Shutdown the authentication service.
 *   The initialization parameters are passed in
 *   again.
 *
 * ARGUMENTS
 *   [in] argc   The count of arguments to the service.
 *   [in] argv   The array of argument strings.
 *
 * RETURN VALUE
 *   XAUTHN_S_COMPLETE
 */
xauthn_status_t
xauthn_shutdown(
    int      argc,      /* in */
    const char **argv  /* in */
)
{
    return XAUTHN_S_COMPLETE;
}

/*
 * FUNCTION NAME
 *   xauthn_authenticate
 *
 * DESCRIPTION
 *   Examine the received user authentication information, and generate a
 *   client identity. The received information will vary depending on the
 *   specified authentication mechanism.
 *
 * ARGUMENTS - IN
 *   authInfo   List of names and set of values containing the user
 *               authentication data. The pdxauthn.h contains all the
 *               possible names that could be in this list. The actual
 *               list of names received will depend on the specified
 *               authentication mechanism.
 *
 * ARGUMENTS - OUT
 *   identity   Pointer to the resulted client identity.
 *
 *   st        Set to XAUTHN_S_COMPLETE, or to an error status indicating

```

```

*           the nature of the failure.
*/
xauthn_status_t
xauthn_authenticate(
    xnvlst_t      *authnInfo,
    xauthn_identity_t *ident
)
{
    char **name = 0;

    printf("=====\n");
    printf("Mapping to soapclient\n");
    printf("=====\n");

    /* This is being used with Active Directory - so use uraf name */
    name = &ident->prin.data.oraf_name;
    ident->prin.prin_type = XAUTHN_PRIN_TYPE_URAF;

    /* set the username to soapclient */
    *name = (char *) strdup( "soapclient" );

    return XAUTHN_S_COMPLETE;
}

xauthn_status_t
xauthn_change_password(
    xnvlst_t      *authnInfo
)
{
    return XAUTHN_S_COMPLETE;
}

```

Use case 2 - WS-Federation

This chapter presents use case 2, a many-to-one federation example utilizing the WS-Federation single sign-on protocol. In this scenario BigCorp is the identity provider and RBTelco is the service provider. RBTelco is providing a telephone conference booking service, which is open to all employees of BigCorp.

The nature of the authentication to RBTelco is many-to-one in that all BigCorp employees are mapped (during the single sign-on) to just one Access Manager account at RBTelco, called `bigcorp_guest`. For audit purposes (to know who actually booked the telephone conference), we carry the BigCorp e-mail address of the actual BigCorp user as an extended attribute in the session credential for `bigcorp_guest`. For personalization of the display at RBTelco, we also carry a display name sent from BigCorp as an extended attribute in the `bigcorp_guest` credential.

8.1 Scenario details

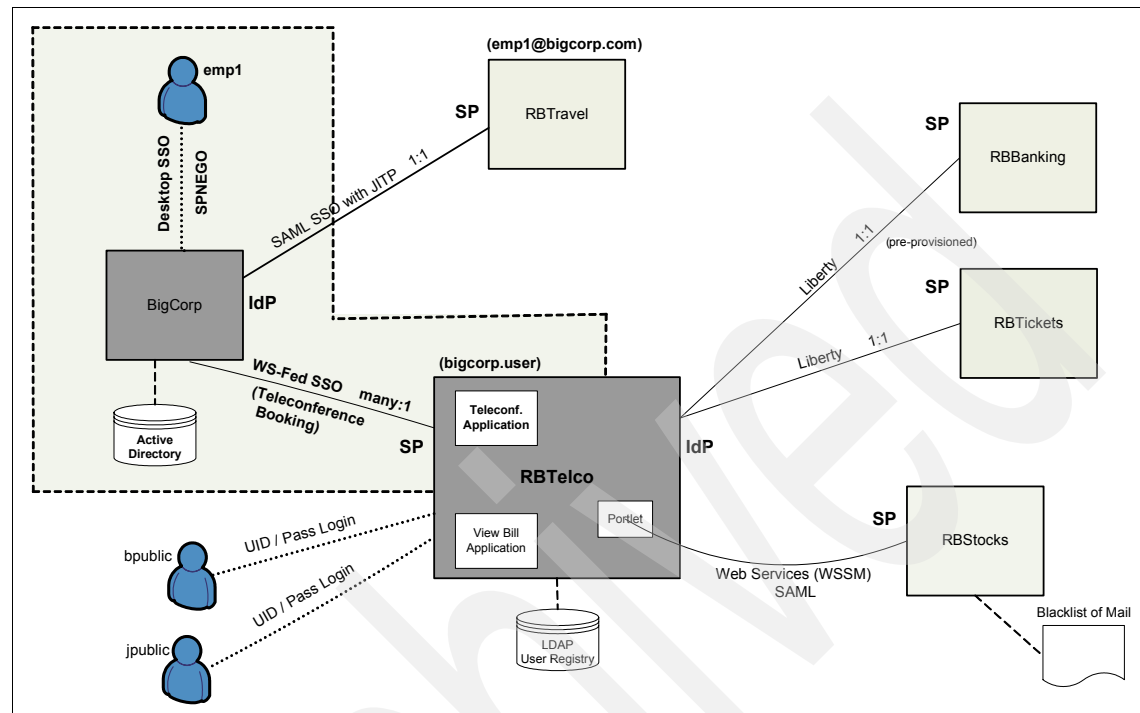


Figure 8-1 Use case 2 logical architecture

We focus on Employee One (emp1) of BigCorp, who logs into his desktop and opens a Web browser with the BigCorp intranet portal. While doing so he automatically gets signed in by the integrated Desktop Single-Sign-On using SPNEGO. Using the portal integrated link to the RBTelco personal tools (which includes the telephone conference booking application), he will be automatically signed in to RBTelco as the user bigcorp_guest.

The components and actors that are present in this use case are highlighted by the grey box in the upper left corner of the diagram shown in Figure 8-1, “Use case 2 logical architecture” on page 220.

8.2 Contract

The very first step in setting up a relation between an IdP and SP is to clarify the technical details of how and what data will be exchanged.

The exchange of intercompany data is a very sensitive issue and will be influenced by many factors before a contract between an identity and a service provider can be signed. We will just stick with the technical facts to ease the understanding of this part and to have the drivers for the federation configuration.

BigCorp and RBTelco have agreed to federate identities using the WS-Federation passive requestor profile. The WS-Federation single sign-on payload will carry a digitally signed SAML 1.1 assertion. The partners will not support single sign out in this case, as this is not strictly possible at BigCorp since the users have desktop single sign-on and will be automatically signed back into the BigCorp portal on their next request. Instead, RBTelco will provide a link to log out the user out from their site only.

The digitally signed SAML assertion will contain the user's e-mail address as the Subject's NameIdentifier. The attribute list of the SAML assertion will contain the display name of the user for personalization of the display at RBTelco. Example 8-1 shows an example SAML assertion, which shows the full format, including attribute name spaces.

Example 8-1 Sample SAML Assertion from BigCorp to RBTelco

```
<saml:Assertion
AssertionID="Assertion-uuid203f1557-0105-f23c-5b82-8ce3efd72411"
IssueInstant="2005-07-16T15:24:29Z"
Issuer="https://www.bigcorp.com/ITFIM/sps/wsfed/wsf" MajorVersion="1"
MinorVersion="1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
  <saml:Conditions NotBefore="2005-07-16T15:14:29Z"
NotOnOrAfter="2005-07-16T15:34:29Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>
        https://www.rbtelco.com/ITFIM/sps/wsfed/wsf
      </saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement AuthenticationInstant="2005-07-16T15:24:29Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        emp1@bigcorp.com
      </saml:NameIdentifier>
    </saml:Subject>
  </saml:AuthenticationStatement>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
```

```

        emp1@bigcorp.com
    </saml:NameIdentifier>
</saml:Subject>
    <saml:Attribute AttributeName="cn"
AttributeNamespace="http://www.bigcorp.com/cn">
    <saml:AttributeValue>
        Employee One
    </saml:AttributeValue>
    </saml:Attribute>
</saml:AttributeStatement>
    <ds:Signature Id="uuid203f1582-0105-efbb-6039-8ce3efd72411"
xmlns:ds="http://www.w3.org/2000/09/xmlsig#">
    <ds:SignedInfo>
        <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#rsa-sha1" />
        <ds:Reference
URI="#Assertion-uuid203f1557-0105-f23c-5b82-8ce3efd72411">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-signature" />
                <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                    <xc14n:InclusiveNamespaces PrefixList="saml ds"
xmlns:xc14n="http://www.w3.org/2001/10/xml-exc-c14n#" />
                </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#sha1" />
            <ds:DigestValue>
                sWS4qUyQXSgMRHM62ADxLHGfFD4=
            </ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
        ....encoded data snipped for readability....
    </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
                ....encoded data snipped for readability....
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</ds:Signature>
</saml:Assertion>

```

8.3 User experience

This section covers both the single sign-on user experience and the logout experience from RBTelco.

8.3.1 Single sign-on user experience

We pick up the single sign-on user experience at BigCorp, just after a user has logged into his desktop and opened his browser to the BigCorp portal page. Since BigCorp employees have SPNEGO authentication to WebSEAL, no explicit authentication is required beyond the desktop login to the Windows domain. More information about the SPNEGO authentication is available in Chapter 7, “Use case 1 - SAML/JITP” on page 193.

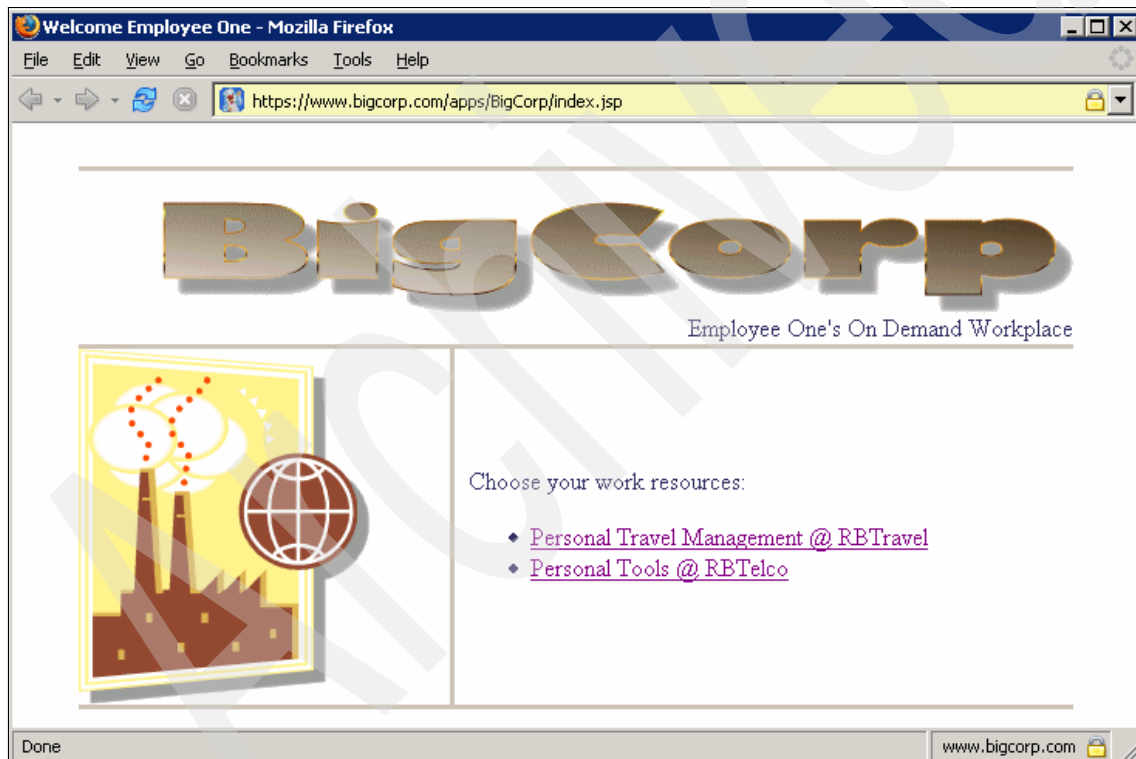


Figure 8-2 BigCorp Portal Intranet page

Figure 8-2 shows the BigCorp Portal Page, with the link to RBTelco.

The user selects the Personal Tools @ RBTelco link. Figure 8-3 shows a page that will briefly be seen by the user as the single sign-on data is automatically posted to RBTelco.

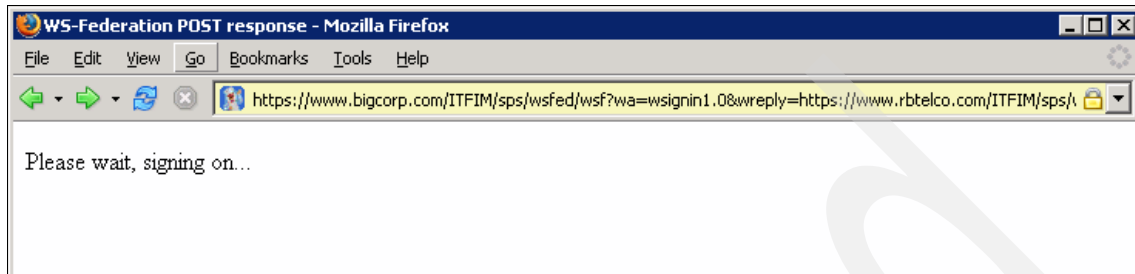


Figure 8-3 Sign-on page from BigCorp

Employee One is then automatically logged into RBTelco. As with most security demonstrations, the user experience is quite unspectacular and quick; but after all, that is the whole point. Figure 8-4 shows the RBTelco portal page after sign-on is complete.

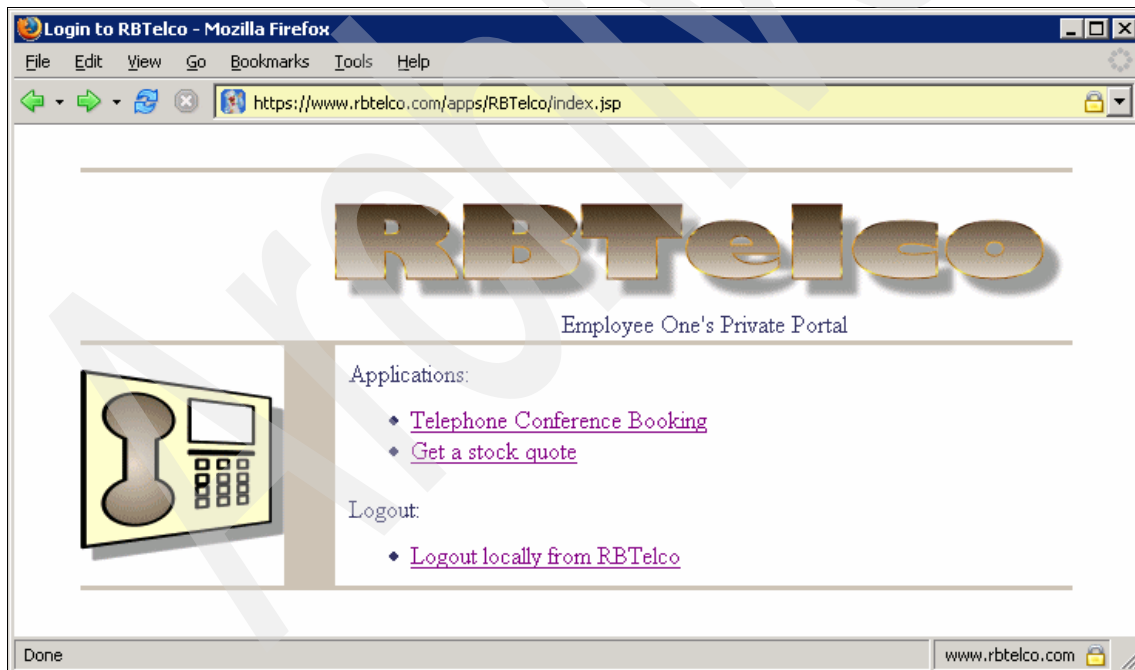


Figure 8-4 RBTelco portal page after single sign-in from BigCorp

The user then selects the telephone conference booking link and sees the booking page shown in Figure 8-5. Notice that it is partner-branded with the bigcorp partner name in the title:

Book your telephone conference for BigCorp here:

This branding is accomplished by an HTTP header that is sent to the application and stored in the bigcorp_guest credential during single sign-on.

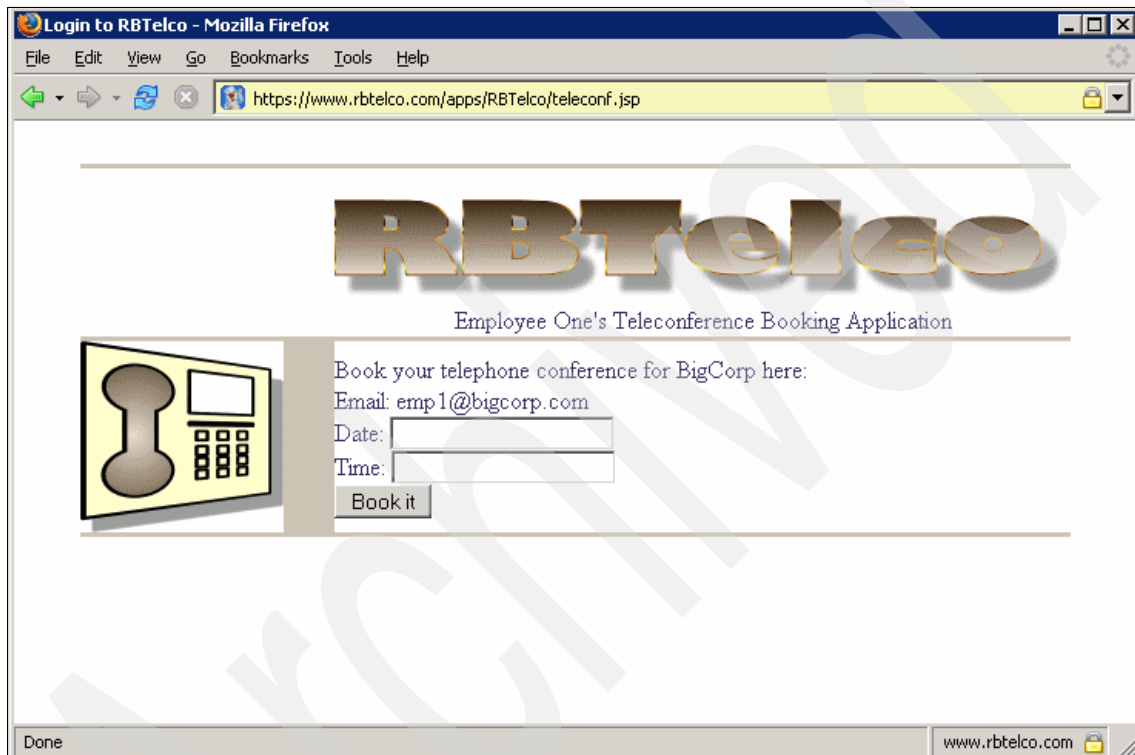


Figure 8-5 RBTelco personalized and branded (for BigCorp) teleconference booking page

8.3.2 Sign-off user experience

After completing booking of the telephone conference, Employee One can choose to log out of the RBTelco Web site. In a typical WS-Federation scenario you would provide a link for single sign-off where the user could be logged out of both the identity provider and service provider simultaneously. If we try to do that with BigCorp, the single sign-off will fail because the user's browser is configured for desktop single sign-on, and the user is automatically re authenticated on their next request. Instead we provide only the regular Access Manager/WebSEAL

logout link on RBTelco, which logs the user out from just the service provider Web site.

Figure 8-6 shows the user at the RBTelco portal page, with a logout link.

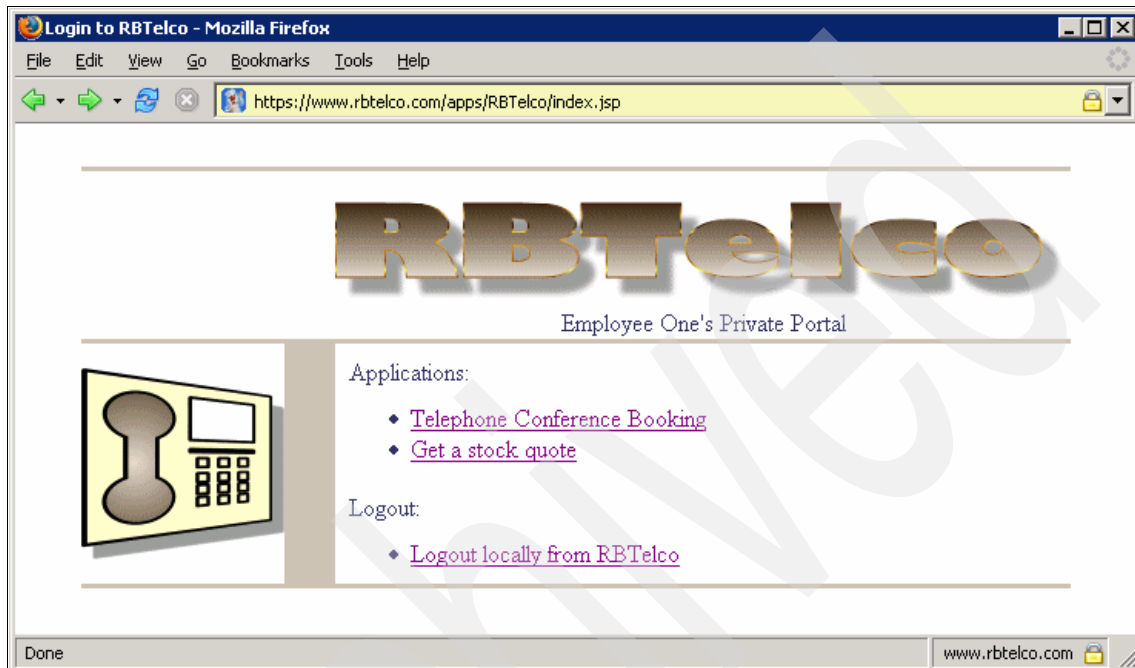


Figure 8-6 RBTelco Page with logout link

After clicking the logout link, Figure 8-7 on page 227 shows the result of the logout. This page is simply a template in WebSEAL that has been customized, since otherwise it would show the logout for the user `bigcorp_guest`, which is the name of the actual Access Manager user in RBTelco for this session. We do not want this displayed to BigCorp employees.

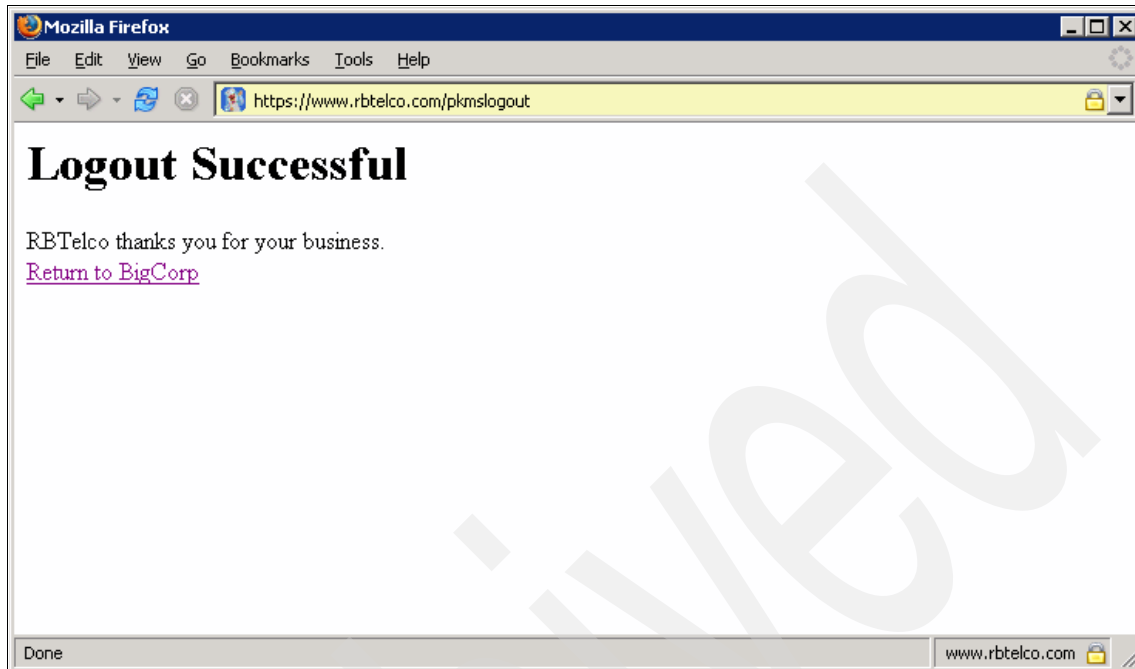


Figure 8-7 Logout from RBTelco

8.4 Functionality

Let us take a closer look at the single sign-on technology used in this scenario.

8.4.1 Single sign-on - WS-Federation

The single sign-on technology used between BigCorp and RBTelco is the WS-Federation passive requestor profile. The detailed specification of this profile can be found at:

<http://www-128.ibm.com/developerworks/webservices/library/ws-fedpass/>

While the specification does not put specific restrictions on the format of the single sign-on token being used, all inter-operability efforts to date have been with SAML 1.1 assertions, and that is the token type used in this use case.

8.5 Partners involved

The corporations involved in this use case are BigCorp and RBTelco.

8.5.1 BigCorp

BigCorp was introduced earlier in use case 1. For the purposes of this use case, BigCorp outsources telephone conference scheduling and management to RBTelco. BigCorp provides a single sign-on user experience for its employees when they go to book teleconferences at RBTelco.

8.5.2 RBTelco

RBTelco is a large provider of telephone and teleconference services. For the purposes of this use case, a single sign-on service is provided for logging into its telephone conference booking application. RBTelco provides a branded and personalized look and feel to the teleconference application for each business partner. This use case only shows one business partner (BigCorp); however, the concept scales to any number of customers.

8.6 Interaction description

The interaction for the WS-Federation single sign-on is shown in Figure 8-8. For those familiar with the SAML single sign-on protocols, this is very similar to a SAML browser-post. A detailed description of the interaction follows the figure.

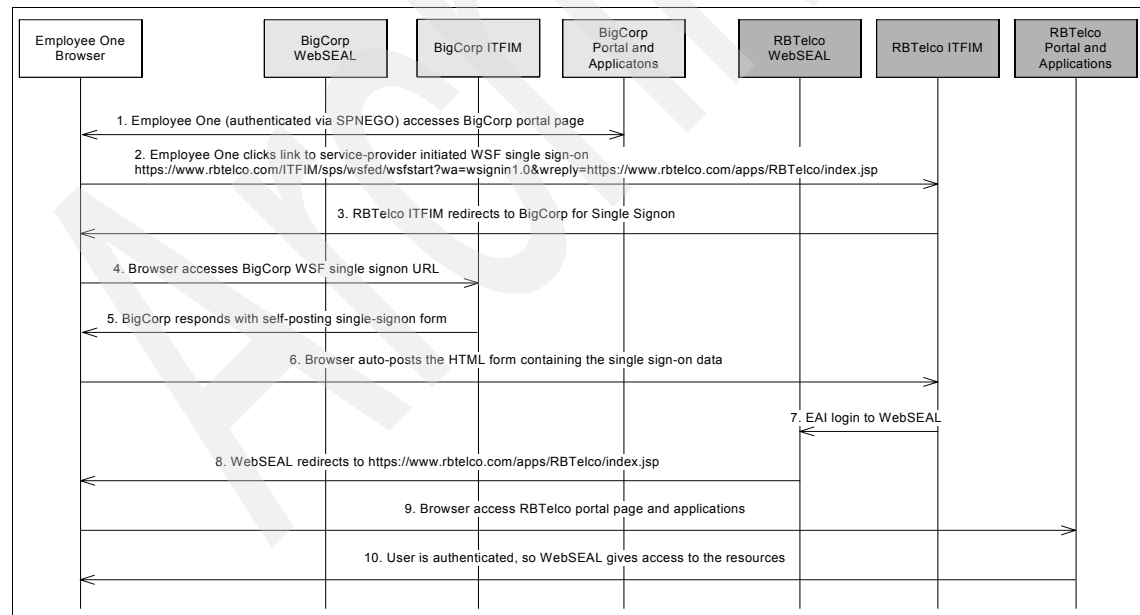


Figure 8-8 Interaction diagram for WS-Federation login

A detailed description of interaction in Figure 8-8 on page 228 follows:

1. Employee One has authenticated to BigCorp via SPNEGO desktop single sign-on and accesses his portal page.
2. Employee One clicks a link to the RBTelco teleconference booking application. This URL is:

```
https://www.rbtelco.com/ITFIM/sps/wsfed/wsfstart?wa=wsignin1.0&wreply=https://www.rbtelco.com/apps/RBTelco/index.jsp
```

This URL will set up some session information for Tivoli Federated Identity Manager on the service provider, and then redirect the user back to the identity provider partner for login.

3. The browser is sent a redirect response to BigCorp's single sign-on URL.

The browser follows the redirect to initiate the WS-Federation single sign-on. This URL is:

```
https://www.bigcorp.com/ITFIM/sps/wsfed/wsf?wa=wsignin1.0&wreply=https://www.rbtelco.com/ITFIM/sps/wsfed/wsf&wctx=https://www.rbtelco.com/apps/RBTelco/index.jsp&wct=2005-07-16T20:34:22Z&wrealm=https://www.rbtelco.com/ITFIM/sps/wsfed/wsf
```

A detailed explanation of these command-line parameters will help with understanding both the configuration and the mechanics of the single sign-on process.

Parameter	Value	Explanation
wa	wsignin1.0	Mandatory parameter to indicate this is a sign-on action
wreply	https://www.rbtelco.com/ITFIM/sps/wsfed/wsf	Indicates URL that RBTelco would like the sign-on request sent to. Tivoli Federated Identity Manager at BigCorp will actually ignore this parameter for security reasons, and always auto-post the sign-on data to a URL in its partner configuration.
wctx	https://www.rbtelco.com/apps/RBTelco/index.jsp	An opaque context parameter that the IdP (BigCorp) should ignore and return unmodified. Tivoli Federated Identity Manager as a service provider actually uses it to carry the destination URL to direct the user to after sign-in.
wct	2005-07-16T20:34:22Z	A UTC time parameter that may optionally (by configuration) be checked to be recent by the IdP.

Parameter	Value	Explanation
wtrealm	https://www.rbtelco.com/ITFIM/sps/wsfed/wsf	A realm identifier to indicate to the IdP from which SP this request is coming.

4. BigCorp's Tivoli Federated Identity Manager generates a self-posting HTML form containing the single sign-on data. Javascript is used to automatically post the form. The Tivoli Federated Identity Manager processing converts the user's Access Manager Credential (passed to Tivoli Federated Identity Manager in the iv-creds header) to a SAML assertion using the Tivoli Federated Identity Manager trust service. A mapping rule, as shown in Example B-11 on page 410, is used to achieve the identity mapping required to generate the SAML assertion in the single sign-on payload.
5. The browser posts the form to the RBTelco's WS-Federation endpoint. The contents of a sample HTML page containing the self-posting form can be seen in Example 8-2.

Example 8-2 Example self-posting HTML form for WS-Federation single sign-on

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>WS-Federation POST response </title>
  </head>
  <body>
    <form method="post"
action="https://www.rbtelco.com/ITFIM/sps/wsfed/wsf">
      <p>
        <input type="hidden" name="wct" value="2005-07-16T21:35:34Z" />
        <input type="hidden" name="wctx"
value="https://www.rbtelco.com/apps/RBTelco/index.jsp" />
        <input type="hidden" name="wresult"
value="&lt;wst:RequestSecurityTokenResponse
xmlns:soapenc=&quot;http://schemas.xmlsoap.org/soap/encoding/&quot;
xmlns:soapenv=&quot;http://schemas.xmlsoap.org/soap/envelope/&quot;
xmlns:wst=&quot;http://schemas.xmlsoap.org/ws/2005/02/trust&quot;
xmlns:wsu=&quot;http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecur
ity-utility-1.0.xsd&quot;
xmlns:xsd=&quot;http://www.w3.org/2001/XMLSchema&quot;
xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
wsu:Id=&quot;uuid2192d263-0105-ec68-dd61-8ce3efd72411&quot;;&gt;&lt;wst:Renewing
Allow=&quot;true&quot;
OK=&quot;false&quot;;&gt;&lt;/wst:Renewing&gt;&lt;wst:KeySize&gt;0&lt;/wst:KeySi
ze&gt;&lt;wst:Forwardable&gt;true&lt;/wst:Forwardable&gt;&lt;wst:Delegatable&gt;
false&lt;/wst:Delegatable&gt;&lt;wst:RequestedTokenReference&gt;&lt;wss:KeyIde
ntifier xmlns:saml=&quot;urn:oasis:names:tc:SAML:1.0:assertion&quot;
```

xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-seext-1.0.xsd";
 ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID";
 Assertion-uuid2192d1ec-0105-f787-5716-8ce3efd72411;
 /wss:KeyIdentifier;
 /wst:RequestedTokenReference;
 /wst:RequestedSecurityToken;
 saml:Assertion
 xmlns:ds="http://www.w3.org/2000/09/xmldsig";
 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion";
 AssertionID="Assertion-uuid2192d1ec-0105-f787-5716-8ce3efd72411";
 IssueInstant="2005-07-16T21:35:34Z";
 Issuer="https://www.bigcorp.com/ITFIM/sps/wsfed/wsf";
 MajorVersion="1";
 MinorVersion="1";
 saml:Conditions
 NotBefore="2005-07-16T21:25:34Z";
 NotOnOrAfter="2005-07-16T21:45:34Z";
 saml:AudienceRestrictionCondition;
 saml:Audience="https://www.rbteleco.com/ITFIM/sps/wsfed/wsf/saml:Audience";
 saml:AudienceRestrictionCondition;
 saml:Condition
 saml:AuthenticationStatement
 AuthenticationInstant="2005-07-16T21:35:34Z";
 AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password";
 saml:Subject;
 saml:NameIdentifier
 Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress";
 email@bigcorp.com;
 /saml:NameIdentifier;
 /saml:Subject;
 saml:AuthenticationStatement;
 saml:AttributeStatement;
 saml:Subject;
 saml:NameIdentifier
 Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress";
 email@bigcorp.com;
 /saml:NameIdentifier;
 /saml:Subject;
 saml:Attribute
 AttributeName="cn";
 AttributeNamespace="http://www.bigcorp.com/cn";
 saml:AttributeValue;
 Employee
 /saml:AttributeValue;
 /saml:Attribute;
 /saml:AttributeStatement;
 ds:Signature
 Id="uuid2192d219-0105-fecc-a844-8ce3efd72411";
 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-uuid2192d1ec-0105-f787-5716-8ce3efd72411";
 ds:Transform
 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature";
 ds:Transform;
 ds:Transform
 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n";
 /ds:Transform;
 ds:Transform
 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n";
 /ds:Transform;
 ds:Transform
 PrefixList="saml";
 /ds:Transform;
 ds:Transform;
 ds:DigestMethod
 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1";
 ds:DigestMethod;
 ds:DigestValue;
 FldzzUWLgfWcY1ODwdCNBr5ee0o;
 /ds:DigestValue;
 ds:Reference;
 ds:SignedInfo;
 ds:SignatureValue;
 Eqb+1ihx

```

AcYr/4amPYlzl1abribdanit5RhhbPKpjQKlBxwSp8VrJQrCl+8PtDecpWKrw6InLxiC4f7av4p010rz
ThXWypqm19Gp0deSlQJ0xQt+jK48Z7txv/3s6zPQbga/VSIvcXiuKtjcmhUS1SO6McIKOHbmrIDc2oc
idjOM=&lt;&lt;/ds:SignatureValue&gt;&lt;&lt;/ds:KeyInfo&gt;&lt;&lt;/ds:X509Data&gt;&lt;&lt;/ds:X50
9Certificate&gt;&lt;MIICqTCCAkgAwIBAgIBEDANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQDEEXNmaW0
ucmVkyM9vay5pYm0uY29tMQswCQYDVQGEwJVUzEMMAoGA1UEChMDSUJNMB4XDTE1MDYwMTIyNTAyOF
oXDEwMDYxNzIyNTAyOFowRTEkMCIgA1UEAxQbYm1nY29ycF9yYnR1bG9uLmJpZ2NvcnAuY29tMQswC
QYDVQGEwJVUzEQMA4GA1UEChMHQm1nQ29ycDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAmsBs
KjCVCq+Uo0G4YjRb3kExfMoFMXv9Y9ypLpBLqcXSy53u+JpCb10ieNzgwP42zhdrxaMSKs6i1LdMfC
rsTd+WtNZ1L1rwpU/eFIu7kCscroAW+HywXrSYQjMPSkn/tUKu+LQeP1TDbAP+55Q8GzDEDcKTYvhI
9TtYN+V+UCAwEAa0BtDCBsTAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBb8Te2jgjl0wmvwLxpqTiiZ
IpDoDBhBgNVHSMWjBYgBRA0z474m+/yN+mhtBtCGxeD8uRmaE9pDswOTEcMBoGA1UEAxMTZm1tLnJl
ZGJvb2suaWJtLmNvbTElMAkGA1UEBhMCVVMxDDAKBgNVBAoTA01CTYIBATALBgNVHQ8EBAMCBLawEgY
JYIZIAyb4QgENBAUWA2hwaDANBgkqhkiG9w0BAQQFAAOBgQAt6QtMH29HVBkS/gaE49yRvKm7EtksM9
+bPXIszbqg8n9Fj1ftPH80yH/AIONNqvyz4ambt1gIkiKjXIzdYrnFknjESWi6i7uoT8us8D+U0e0qF
12wAsHkGuyy1ff32MaYzNp1EbftTQ1oTKi6K1AAtkr22A2Fi1SGE7uLfiXnsQ==&lt;&lt;/ds:X509Cert
ificate&gt;&lt;&lt;/ds:X509Data&gt;&lt;&lt;/ds:KeyInfo&gt;&lt;&lt;/ds:Signature&gt;&lt;&lt;/sam
l:Assertion&gt;&lt;&lt;/wst:RequestedSecurityToken&gt;&lt;&lt;/wst:Status&gt;&lt;&lt;/wst:Cod
e&gt;&lt;http://schemas.xmlsoap.org/ws/2005/02/security/trust/status/valid&lt;&lt;/wst:
Code&gt;&lt;&lt;/wst:Status&gt;&lt;&lt;/wst:RequestSecurityTokenResponse&gt;&lt;&lt;/>
    <input type="hidden" name="wa" value="wsignin1.0" />
    <noscript>
    <button type="submit">POST</button> <!-- included for
requestors that do not support javascript -->
    </noscript>
  </p>
</form>
<script type="text/javascript">
  var signonText = 'Please wait, signing on...';
  document.write(signonText);
  setTimeout('document.forms[0].submit()', 0);
</script>
</body>
<html>

```

An explanation of the FORM fields in this self-posting form will help explain how the single sign-in is completed.

Parameter	Value	Explanation
wa	wsignin1.0	Mandatory parameter to indicate this is a sign-on action.
wctx	https://www.rbtelco.com/apps/RBTelco/index.jsp	The same opaque context parameter passed from RBTelco previously.

Parameter	Value	Explanation
wct	2005-07-16T21:35:34Z	A UTC time parameter that may optionally (by configuration) be checked to be recent by the SP.
wresult	Request security token response (not repeated in table for brevity)	This contains a trust service response message, which is a wrapper around the signed SAML assertion for the user. Note that the whole field is encoded so that XML tags do not confuse the browser.

6. Tivoli Federated Identity Manager at RBTelco validates the sign-in request and performs identity mapping. The mapping rule, as shown in Example B-12 on page 412, is used to convert the SAML assertion to the Access Manager user `bigcorp_guest` while keeping the user's home company, actual e-mail address, and display name in extended attributes in the credential.

After the credential is generated, it is sent back to WebSEAL in special HTTP headers as part of an EAI authentication. "External Authentication Interface" on page 370 explains EAI authentication in more detail.

7. WebSEAL also receives from Tivoli Federated Identity Manager the URL to redirect the browser to after authentication. This is the `wreply` parameter from step 2. WebSEAL sends a session cookie to the browser, along with a redirect to this URL.
8. The browser is now authenticated and accesses protected resources at RBTelco.
9. RBTelco honors the request for the protected resources.

8.7 Configuration data

This section discusses the configuration information used for both the identity provider (BigCorp) and the service provider (RBTelco) for this federation.

As a starting point we assume that Tivoli Federated Identity Management is installed, and the runtime deployed and configured. Additionally, the junction between WebSEAL and Tivoli Federated Identity Management is assumed to be configured.

The following references will assist with the installation and configuration of Tivoli Federated Identity Management:

- *IBM Tivoli Identity Manager Installation Guide Version 6.0*, GC32-1667-00, discusses the installation of Tivoli Federated Identity Management.

- ▶ *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00, contains basic information about configuring the Tivoli Federated Identity Management runtime and information on configuring federations.
- ▶ Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, contains information about configuring Tivoli Federated Identity Management for use with WebSEAL.

8.7.1 Identity provider configuration at BigCorp

Configuring WS Federation at BigCorp consists of the following tasks:

- ▶ Importing BigCorp signing keys
- ▶ Configuring Tivoli Federated Identity Management for WS-Federation as an identity provider
- ▶ Configuring a service provider partner for RBTelco
- ▶ Configuring Access Manager policy for the federation URLs

Importing BigCorp keys

Appendix C, “Keys and certificates” on page 425, contains information about the key strategy used for all use cases. In particular, note that for this federation configuration the `bigcorp-signing.jks` key file was imported into Tivoli Federated Identity Management. This contains the signing key used to sign the SAML assertion sent to RBTelco.

Configuring BigCorp as a WS-Federation identity provider

Detailed information about configuring a WS-Federation as an identity provider is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00. This section discusses the specific configuration parameters used for BigCorp.

Table 8-1 contains specific configuration data used for configuring the identity provider WS-Federation at BigCorp.

Table 8-1 Configuration information for BigCorp as WS-Federation identity provider

Field	Value used in this use case
Federation name	wsf.
Identify your role	Identity provider.
Identity Provider Company Name	BigCorp.
Protocol for federation	WS-Federation Passive Profile.

Field	Value used in this use case
Point of contact server configuration	https://www.bigcorp.com/ITFIM/sps
Token Module	Select the default WS-Federation token type for SAML 1.1.
Security Token Configuration	SAML Assertion validity period: 60 seconds before and 60 seconds after current time. This allows for clock skew differences with the partner.
Identity Mapping	The mapping rule used for mapping the Access Manager credential at BigCorp to a SAML 1.1 assertion is available at Example B-11 on page 410.

Figure 8-9 on page 236 shows the summary page of the identity provider federation configuration at BigCorp.

Federation Properties

*Federation Name
wsfed

My Role: Identity Provider

*Company Name
BigCorp

Company URL
https://www.bigcorp.com/

Contact Person

First Name	Last Name
Bobby	Barramundi

Email Address
barra@bigcorp.com

Phone Number
555-1234

Security Assertion Markup Language (SAML) Module Configuration

*Amount of time the assertion is valid after being issued (seconds)
60

[View/Edit Identity Mapping Rule...](#)

My Single Sign-On Properties

SSO Protocol: WS-Federation Passive Profile

*WS-Federation Realm
https://www.bigcorp.com/ITFIM/sps/wsfed/wsf

*WS-Federation Endpoint
https://www.bigcorp.com/ITFIM/sps/wsfed/wsf

OK Apply Cancel

Figure 8-9 BigCorp WS-Federation identity provider summary page

Note: This figure only shows the validity period configuration for the number of seconds after the current time. At the time of writing this book, the validity period setting for before the current time had not been added to the graphical console.

Configuring a service provider partner for RBTelco

Table 8-2 on page 237 contains specific configuration data used for configuring the RBTelco service provider partner at BigCorp.

Table 8-2 Configuration information for BigCorp as WS-Federation identity provider

Field	Value
Service Provider Company Name	RBTelco
WS-Federation Realm	https://www.rbtelco.com/ITFIM/sps/ws fed/wsf
WS-Federation Endpoint	https://www.rbtelco.com/ITFIM/sps/ws fed/wsf
Maximum Request Lifetime	-1
Key for signing assertions	bigcorp-signing_bigcorp_rbtelco
Mapping rule	No need to provide one at the partner level, since we chose to provide one at the federation level

Figure 8-10 on page 238 shows the summary page of the service provider partner federation configuration at BigCorp.

Partner Properties	
*Company Name	RBTelco
Company URL	https://www.rbtelco.com/
Contact Person	
First Name	Last Name
Timmy	Telecom
Email Address	tt@rbtelco.com
Phone Number	555-4321
Member of Federation:	wsfed
Partner Role:	Service Provider
Status:	Enabled <input type="button" value="Disable"/>
Security Assertion Markup Language (SAML) Module Configuration	
<input checked="" type="checkbox"/>	Enable the Signing of Assertions
Select Key for Signing Assertions	bigcorp-signing_bigcorp_rbtelco
Include the following attribute types (a "*" means include all types)	*
<input type="button" value="View/Edit Identity Mapping Rule..."/>	
Single Sign-On Properties	
*WS-Federation Realm	https://www.rbtelco.com/ITFIM/sps/wsfed/wsf
*WS-Federation Endpoint	https://www.rbtelco.com/ITFIM/sps/wsfed/wsf
Maximum Request Lifetime (in seconds)	-1

Figure 8-10 RBTelco service provider partner summary page at BigCorp

Configuring Access Manager policy at BigCorp

In this case there is actually no specific requirement for Access Manager configuration of the WS-Federation URL at BigCorp. This is because BigCorp's policy is to allow any employee to access the RBTelco-hosted teleconference booking application. That being the case, the default-webseal ACL, which allows access to any authenticated user, is suitable. If only some employees were allowed to access the application (for example, managers), then the following object in BigCorp should be appropriately protected with the Access Manager authorization policy:

```
/WebSEAL/<webseal_server>/ITFIM/sps/wsfed/wsf
```

8.7.2 Service provider configuration at RBTelco

Configuring WS Federation at RBTelco consists of the following tasks:

- ▶ Importing RBTelco partner keys
- ▶ Configuring Tivoli Federated Identity Management for WS-Federation as a service provider
- ▶ Configuring an identity provider partner for BigCorp
- ▶ Configuring Access Manager policy for the federation URLs

Importing RBTelco keys

Appendix C, “Keys and certificates” on page 425, contains information about the key strategy used for all use cases. In particular, note that for this federation configuration the `rbtelco-partners.jks` key file was imported into Tivoli Federated Identity Management. This contains the public certificate used to verify the signature on the SAML assertion sent from BigCorp.

Configuring RBTelco as a WS-Federation service provider

Detailed information about configuring a WS-Federation as a service provider is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00. This section discusses the specific configuration parameters used for RBTelco.

Table 8-3 contains specific configuration data used for configuring the service provider WS-Federation at RBTelco.

Table 8-3 Configuration information for RBTelco as WS-Federation service provider

Field	Value Used in this use case
Federation name	wsf.
Identify your role	Service provider.
Service Provider Company Name	RBTelco.
Protocol for federation	WS-Federation Passive Profile.
Point of contact server configuration	https://www.rbtelco.com/ITFIM/sps
Identity Mapping	The mapping rule used for mapping the SAML assertion from BigCorp to an Access Manager credential at RBTelco is available in Example B-12 on page 412.

Figure 8-11 on page 240 shows the summary page of the service provider federation configuration at RBTelco.

Federation Properties

*Federation Name
wsfed

My Role: Service Provider

*Company Name
RBTelco

Company URL
https://www.rbtelco.com

Contact Person

First Name: Timmy Last Name: Telecom

Email Address: tt@rbtelco.com

Phone Number: 555-4321

[View/Edit Identity Mapping Rule...](#)

My Single Sign-On Properties

SSO Protocol: WS-Federation Passive Profile

*WS-Federation Realm
https://www.rbtelco.com/ITFIM/sps/wsfed/wsf

*WS-Federation Endpoint
https://www.rbtelco.com/ITFIM/sps/wsfed/wsf

OK Apply Cancel

Figure 8-11 RBTelco WS-Federation service provider summary page

Configuring an identity provider partner for BigCorp

Table 8-4 contains specific configuration data used for configuring the BigCorp identity provider partner at RBTelco.

Table 8-4 Configuration information for BigCorp as WS-Federation identity provider

Field	Value
Identity Provider Company Name	BigCorp
WS-Federation Realm	https://www.bigcorp.com/ITFIM/sps/wsfed/wsf
WS-Federation Endpoint	https://www.bigcorp.com/ITFIM/sps/wsfed/wsf
Maximum Request Lifetime	-1
Key for validating signed assertions	rbtelco-partners_bigcorp_rbtelco

Field	Value
Mapping rule	No need to provide one at the partner level, since we chose to provide one at the federation level

Figure 8-12 shows the summary page of the identity provider partner federation configuration at RBTelco.

Partner Properties

*Company Name
BigCorp

Company URL

Contact Person

First Name Last Name

Email Address

Phone Number

Member of Federation: wsfed
Partner Role: Identity Provider
Status: Enabled

Security Assertion Markup Language (SAML) Module Configuration

Enable Signature Validation

*Select Validation Key
rbtelco-partners_bigcorp_rbtelco

Single Sign-On Properties

*WS-Federation Realm
https://www.bigcorp.com/ITFIM/sps/wsfed/wsf

*WS-Federation Endpoint
https://www.bigcorp.com/ITFIM/sps/wsfed/wsf

Maximum Request Lifetime (in seconds)
-1

Figure 8-12 BigCorp identity provider partner summary page at RBTelco

Configuring Access Manager policy at RBTelco

The WS-Federation endpoint at RBTelco is being used to authenticate users to the RBTelco WebSEAL. Consequently, it is necessary to allow unauthenticated

access to this URL. “Access Manager policy for trigger URLs for EAI” on page 376 discusses the need for this, and in our case the federation URL to which the unauthenticated-allowed ACL needs to be applied is:

```
/WebSEAL/<webseal_server>/ITFIM/sps/wsfed/wsf
```

8.8 Assumptions/implementation notes

This section contains use case 2 specific information that may be of interest to the reader.

8.8.1 Understanding the many-to-one user identity mapping

Performing a many-to-one mapping at RBTelco is a very powerful capability that can dramatically reduce the overhead of managing user accounts for every BigCorp employee at RBTelco.

To understand what is actually happening let us first take a look at the Access Manager credential the user started with at BigCorp. Figure 8-13 on page 243 shows a screen capture of the credential at BigCorp. This display comes from the Access Manager **epac** cgi demonstration program, which displays the internal contents of the Access Manager credential in a browser. Notice the two parameters at the end, which were read from this user's active directory entry:

- ▶ tagvalue_activatedir_cn carries the user's display name.
- ▶ tagvalue_activatedir_mail carries the user's e-mail address.

Index	Name	Value(s)
0	AUTHENTICATION_LEVEL	[0]: 0
1	AZN_CRED_AUTHNMECH_INFO	[0]: GSS Authentication
2	AZN_CRED_AUTHZN_ID	[0]: emp1
3	AZN_CRED_AUTH_METHOD	[0]: kerberosv5
4	AZN_CRED_BROWSER_INFO	[0]: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.8) Gecko/20050511 Firefox/1.0.4
5	AZN_CRED_GROUPS	[0]: employees
6	AZN_CRED_GROUP_REGISTRY_IDS	[0]: CN=employees,CN=Users,DC=bigcorp,DC=com
7	AZN_CRED_GROUP_UUIDS	[0]: 6f0f791c-aa49-11d9-a4eb-000c29d2099e
8	AZN_CRED_IP_ADDRESS	[0]: 0x03050309
9	AZN_CRED_MECH_ID	[0]: IV_URAF_V3.0
10	AZN_CRED_PRINCIPAL_DOMAIN	[0]: Default
11	AZN_CRED_PRINCIPAL_NAME	[0]: emp1
12	AZN_CRED_PRINCIPAL_UUID	[0]: 9d5f1ea8-df36-11d9-872b-000c29a951ea
13	AZN_CRED_QOP_INFO	[0]: None
14	AZN_CRED_REGISTRY_ID	[0]: CN=Employee One,CN=Users,DC=bigcorp,DC=com
15	AZN_CRED_USER_INFO	[0]:
16	AZN_CRED_VERSION	[0]: 0x00000510
17	tagvalue_active_dir_cn	[0]: Employee One
18	tagvalue_active_dir_mail	[0]: emp1@bigcorp.com

Figure 8-13 Access Manager credential at BigCorp for Employee One

Now look at the credential at RBTelco, as shown in Figure 8-14 on page 244. Note that the Access Manager user is bigcorp_guest, and that extended attributes that originally came from BigCorp, and were carried in the SAML 1.1 assertion during single sign-on, are in this session credential and are used to carry:

- ▶ Where this user has come from (in tagvalue_fim_partner)
- ▶ The user's display name (in tagvalue_cn)
- ▶ The user's e-mail address (in tagvalue_mail)

These attributes are downstream to the BigCorp portal and applications as HTTP headers using Access Manager WebSEAL's standard tag/value support. You can see them used in the RBTelco portal page and the telephone conference booking application.

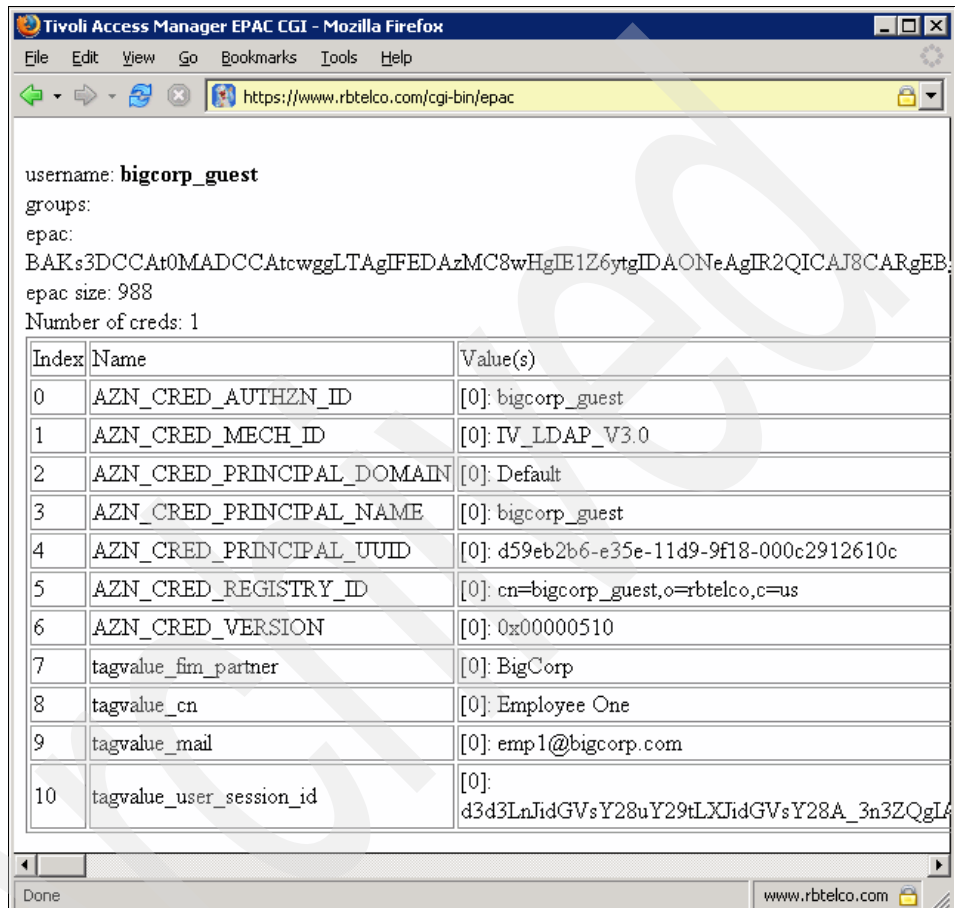


Figure 8-14 Access Manager Credential at RBTelco after WS-Federation sign-in from BigCorp

Use case 3 - Liberty

This chapter presents use case 3, a typical 1:1 user mapping scenario between accounts at RBTelco and its partners, utilizing the Liberty single sign-on protocol. In this scenario RBTelco is the identity provider and RBTickets and RBBanking are the service providers. RBTickets and RBBanking provide value-add services to RBTelco's retail customers (note that these are different from the corporate business partner customers presented in use case 2.)

The nature of the authentication between RBTelco and its partners is 1:1, in that the end user has individual accounts at both RBTelco and at the service provider companies. These accounts are "linked" using a process in Liberty known as federation, so that once federated, the user is able to sign in to the partners by providing authentication credentials only at RBTelco.

One of the features of the Liberty protocol is that no personal information about the user from RBTelco is actually shared with the partner companies. In many real-world scenarios this is enforced for privacy reasons.

9.1 Scenario details

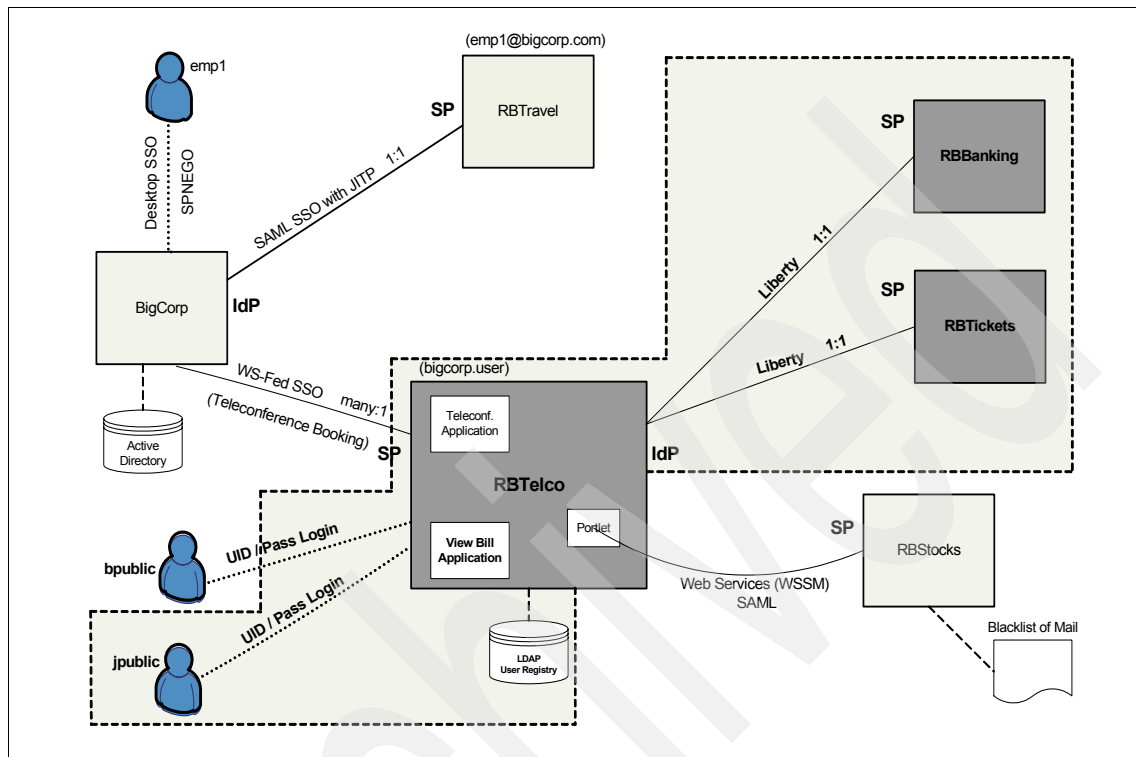


Figure 9-1 Use case 3 logical architecture

Our focus is on John Public, who will use a Web browser with the RBTelco portal. Using the portal's integrated links to the RBBanking and RBTickets servers he is able to link accounts and single sign on to these destinations.

The components and actor that are present in this use case are highlighted by the gray box from the lower left corner up over the upper right of the diagram shown in Figure 9-1, "Use case 3 logical architecture" on page 246.

9.1.1 Contract

RBTelco has a general approach to federate its identities with RBBanking and RBTickets using the Liberty 1.2 protocol. The idea is to serve the customers with a convenient account linking and single sign-on experience within its portal.

RBBanking and RBTickets have agreed with RBTelco to use Liberty Version 1.2 based federated identity management. No personal information will be shared between the companies about the user.

In particular, the following Liberty 1.2 protocols will be supported:

- ▶ Account federation, initiated at the identity provider (RBTelco) using the SOAP/HTTP profile
- ▶ Single sign-on, using the browser-artifact profile
- ▶ Single sign-off using the HTTP redirect profile

Liberty also provides the following protocols that we have chosen not to explore in this use case, simply to keep a more focused view on the common scenario:

- ▶ Defederation - A protocol for unlinking the accounts at the identity provider and service provider
- ▶ Register Name Identifier - A protocol for refreshing the unique identity key shared between the identity provider and service provider

9.1.2 User experience

The following sections illustrate the user experience for the three protocols mentioned above.

Account Linking

The process of account linking requires the end user (John Public in our case) to manually log in to individual accounts at each of the identity provider and service provider partners, and choose to link these accounts. In this scenario we present John Public, and he has the following accounts at each of the companies.

Company	Username
RBTelco	jpublic
RBBanking	jp
RBTickets	johnp

Figure 9-2 on page 248 shows the RBTelco portal page with John Public logged in using his jpublic account. For the purposes of this demonstration, he is already linked with his account at RBBanking, and this scenario walks through the linking of his account with RBTickets.



Figure 9-2 RBTelco portal prior to linking accounts with RBTickets

John then clicks the Federate link for RBTickets, and is shown the login page at RBTickets. This is because he had not yet authenticated to RBTickets. At RBTickets, he logs in using his johnp account, as shown in Figure 9-3 on page 249.



Figure 9-3 Login page at RBTickets during account linking

Following login at RBTickets, and per federation configuration at RBTelco, John is required to confirm his consent to federate these accounts. This step is part of the Liberty process, and can be configured on or off. Figure 9-4 shows the Tivoli Federated Identity Manager default page for federation consent. This can be customized for a more meaningful look and feel.

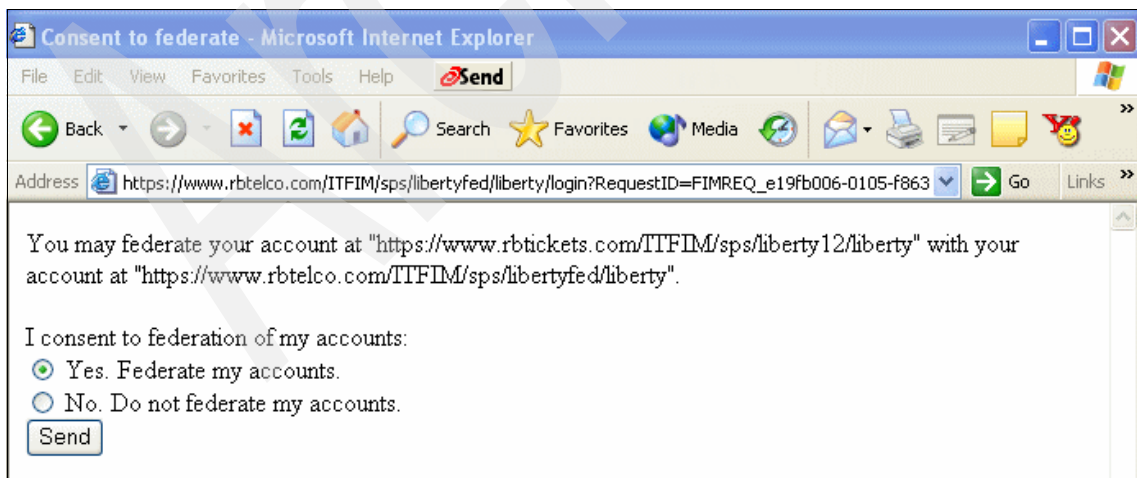


Figure 9-4 Consent to federate during account linking

After accepting the consent to federate page, John's accounts at RBTelco and RBTickets are automatically linked, and he is shown the RBTickets portal page, as shown in Figure 9-5.

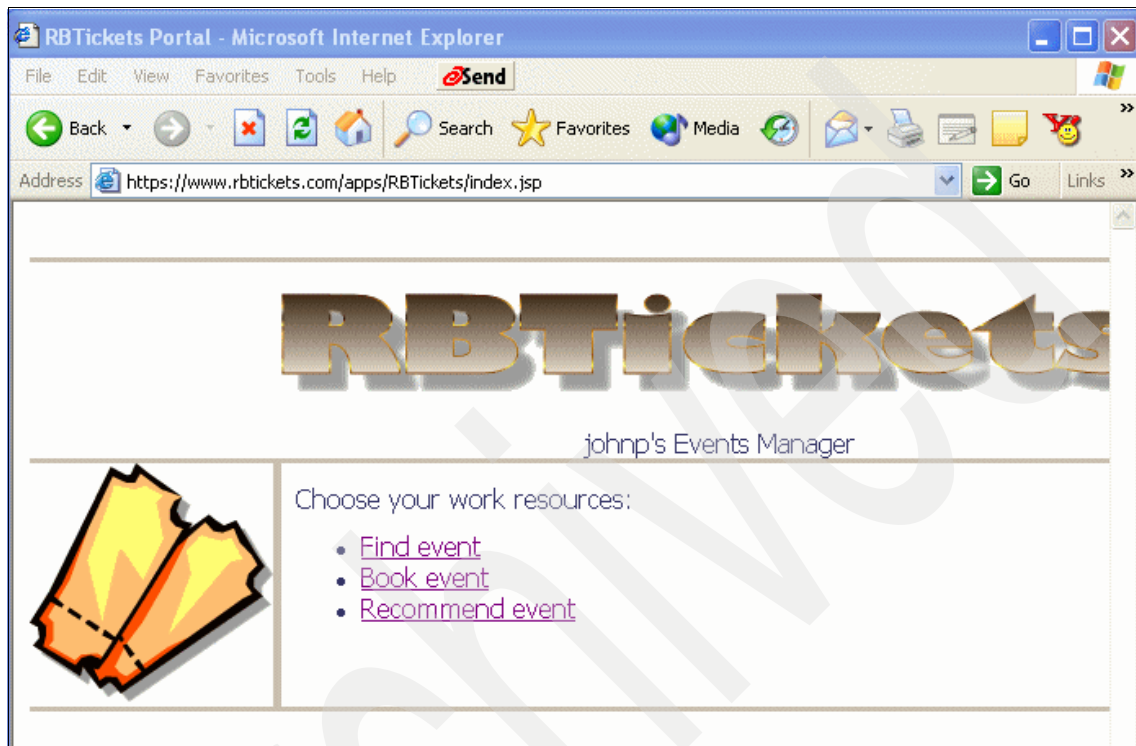


Figure 9-5 RBTickets portal page after account linking complete

Single sign-on

The single sign-on experience is the primary reason for using any of the federation protocols, and provides the convenient service of requiring login to only the identity provider site. Simply by clicking a Web link, automated login is provided to each of the service provider sites. In this user experience scenario we will illustrate a login to RBTelco, and then show one-click automatic logins to each of RBBanking and RBTickets. This scenario can only be run after John has linked accounts to both RBBanking and RBTickets.

Figure 9-6 on page 251 shows the RBTelco portal after John has logged in using his johnp account. Note that this page looks different from the RBTelco portal page shown in Figure 9-2 on page 248. The RBTelco portal page is rendered with the help of Tivoli Federated Identity Manager InfoService APIs, which let you determine which partners a particular user is federated with. More on this capability is explained in 9.6.1, "InfoService integration" on page 284.



Figure 9-6 RBTelco portal after jpublic login with accounts linked

Simply by clicking the RBBanking link, jpublic is automatically logged into RBBanking as his user ID jp. Similarly, by clicking the RBTickets link from the RBTelco portal, jpublic is automatically signed in to RBTickets as johnp. These screens are shown in Figure 9-7 on page 252 and Figure 9-8 on page 253.

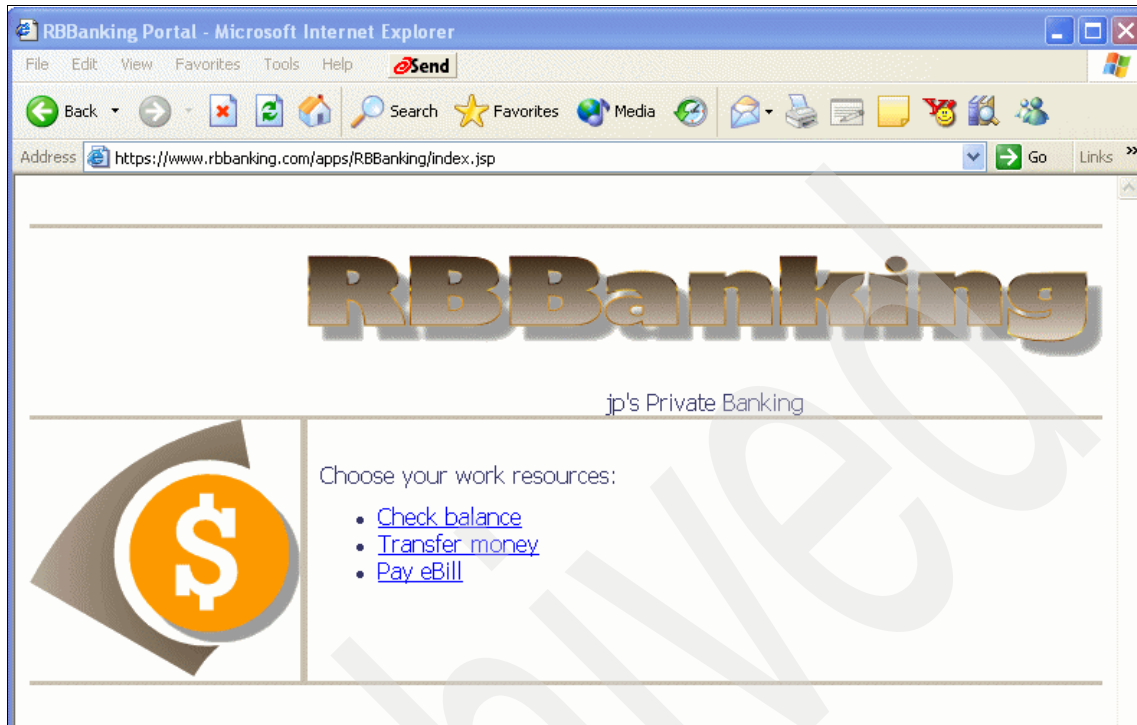


Figure 9-7 RBBanking portal after single sign-on from RBTelco

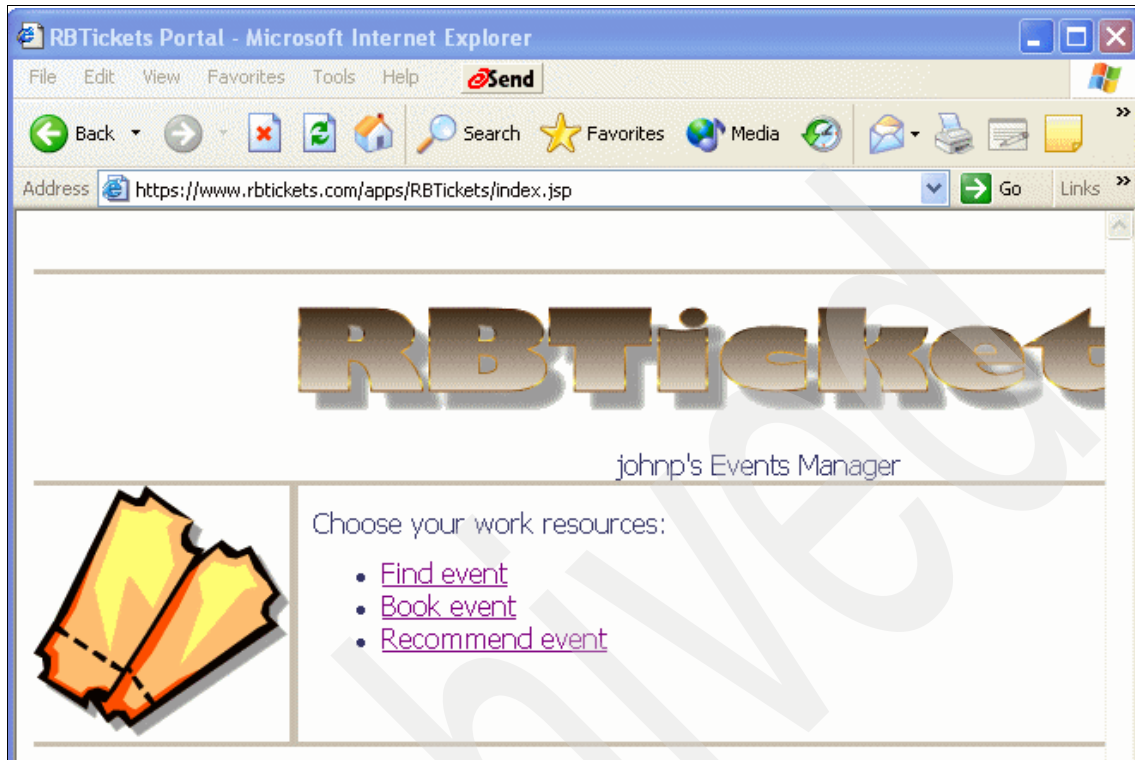


Figure 9-8 RBTickets portal after single sign-on from RBTelco

Single sign-off

The single sign-off protocol (in our case initiated at the identity provider) gives all authenticated parties (the identity provider and any number of service provider partners) the opportunity to destroy session information for the user, thereby facilitating a logout.

While we have only provided the logout button on the RBTelco portal page (due to development time constraints), the logout can be initiated from any service provider or the identity provider. When John accesses the Liberty Logout button from the RBTelco portal, a series of redirects results in him being logged out from each of the service providers, and the identity provider. The final logout result page is shown in Figure 9-9 on page 254. This can be customized for a richer look and feel.

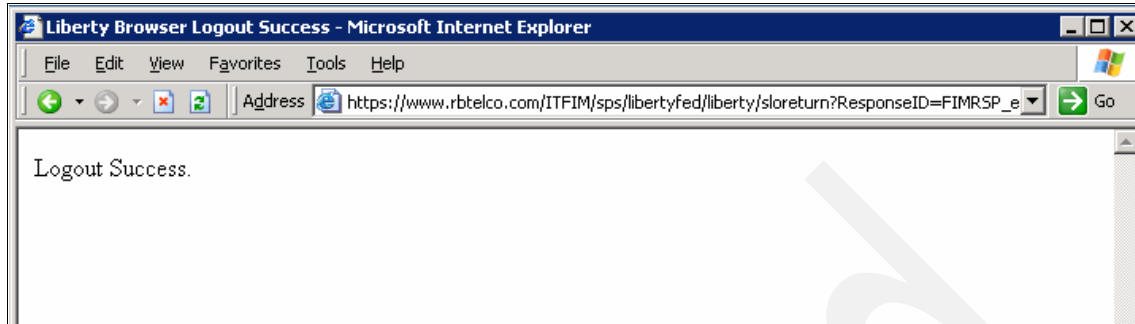


Figure 9-9 Logout success page after initiating http-redirect liberty logout from RBTelco

9.2 Functionality

All functionality described in this use case is part of the Liberty 1.2 specification. We employed a subset of this functionality to implement the use case. The Liberty 1.2 specification is available at:

<http://www.projectliberty.org>

9.3 Partners involved

The corporations involved in this use case are RBTelco and its partners RBanking and RBTickets.

9.3.1 RBTelco

RBTelco has already been introduced in 8.5.2, “RBTelco” on page 228, as a service provider to BigCorp. In this scenario RBTelco acts as an identity provider and enriches its portal with services of partners like RBTickets and RBanking.

9.3.2 RBTickets

RBTickets specialized in event marketing and ticketing. RBTelco offers discounted ticket deals for its retail customers, and a single sign-on experience into the RBTickets site. Customers can use RBTickets to view deals and purchase tickets.

9.3.3 RBanking

RBanking provides automated bill payment services to retail customers of RBTelco. RBTelco provides a single sign-on experience into RBanking to make it more convenient for customers to pay their bills online.

9.4 Interaction description

The sections below show the interaction diagrams for each of the federation, single sign-on, and single sign-off flows.

9.4.1 Liberty account federation

The interaction for federation (account linking) is shown in Figure 9-10. A detailed description of the interaction follows the figure.

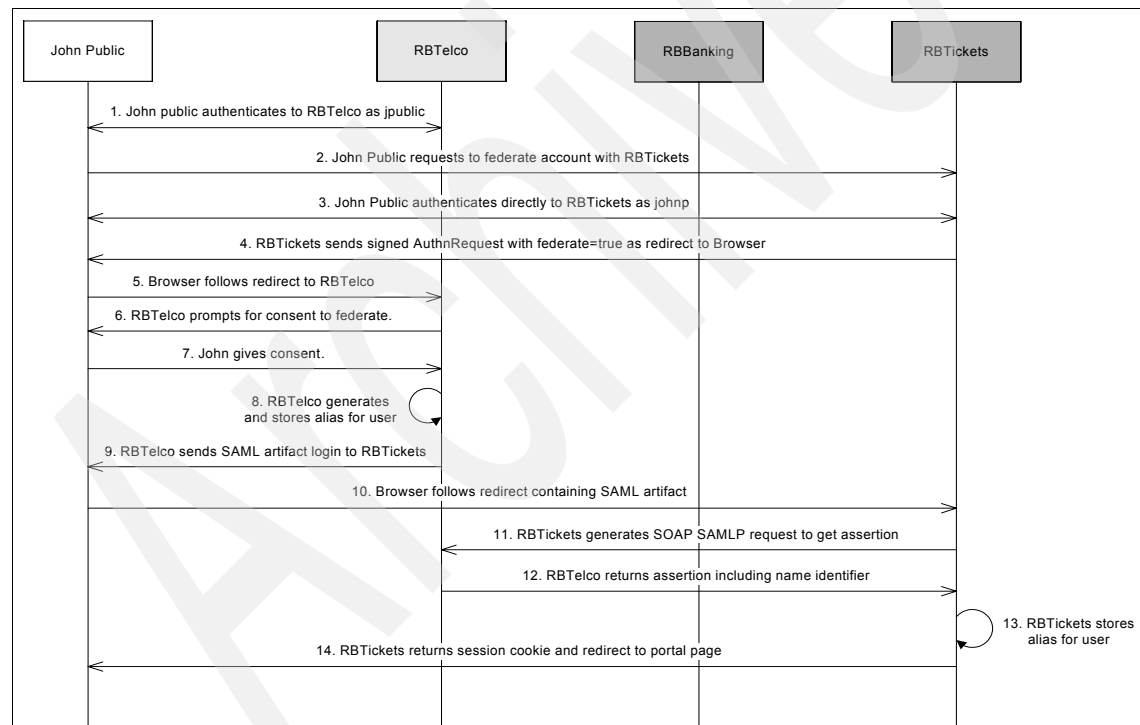


Figure 9-10 Liberty account federation

Detailed description of interaction in Figure 9-10 on page 255:

1. John Public authenticates to RBTelco using his jpublic user name and password.
2. John Public clicks a link to federate his jpublic account with RBTickets. This URL is:

```
https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/login?RelayState=https://www.rbtickets.com&Federate=true
```

Note that this URL is a protected URL on the RBTickets Web site, which will force a login at RBTickets if the user does not already have a session.

3. John Public authenticates with the johnp user name and password before Tivoli Federated Identity Manager at RBTickets processes the federate request.
4. RBTickets generates a redirect to the browser to federate at RBTelco. Since the browser has a session with RBTelco, RBTelco knows who the user is at its site (jpublic), without knowing who the user is at RBTickets. Similarly, RBTickets knows who the user is for RBTickets (johnp), but not for RBTelco. Only the browser user, who has sessions with each site, knows both user IDs. This redirect is signed, and looks like:

```
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/login?RequestID=FIMREQ_e2d7c8b4-0105-ef55-798f-9ab00c5a78ff&MajorVersion=1&MinorVersion=2&IssueInstant=2005-08-23T10:3A17:3A36Z&ProviderID=https%3A%2F%2Fwww.rbtickets.com%2FITFIM%2Fsps%2Fliberty12%2Fliberty&IsPassive=false&NameIDPolicy=federated&ProtocolProfile=http%3A%2F%2Fprojectliberty.org%2Fprofiles%2Fbrws-art&RelayState=uuide2d1974a-0105-fbc6-edde-9ab00c5a78ff&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1&Signature=kNpWtCpFshsEgx3U0ybNt%2BJ395BoGHq%2BjoItsypw4Kzcs%2FD4kpMI1hpYVhvKNMKuhCVqKJNyWB3q%0D%0AWmYPdAinS81EguySx5VthK589Wmt1JPNxTke2b3F4Am1uMb34CyZu0mv0deurkK8JS0Kag0g%2B4Xt%0D%0AT9IcuEeXR%2Bnmv9Icdpc%3D
```

5. The browser follows the redirect and sends the signed login request (with NameIDPolicy set to federated). Note that the ProtocolProfile property is also set to the browser artifact profile. This determines RBTelco's response type in step 8.
6. By way of configuration at RBTelco, John is prompted for consent to federate.
7. John gives his consent to federate.
8. Tivoli Federated Identity Manager at RBTelco generates and stores a name identifier (unique ID) for jpublic.

9. RBTelco also generates a Liberty Assertion for the login, and redirects to RBTickets with the SAML artifact, per the browser-artifact profile. This redirect looks like:


```
https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/login?RelayState=uid
e2d1974a-0105-fbc6-edde-9ab00c5a78ff&SAMLart=AA0%2Fpac7rYy13xjECRYohKvb5qdv
02BbrDK0s9keWMJK9RGyx9TMJWPG
```
10. The browser follows the redirect containing the SAML artifact.
11. RBTickets generates a SAML request to exchange the artifact for an assertion. This request is shown in Example 9-1.
12. RBTelco returns the assertion, which contains the name identifier, as shown in Example 9-2 on page 258. At this point RBTickets still knows who the user really is (from their initial manual login), and now has their unique name identifier.
13. RBTickets stores the name identifier as a mapping to the johnp user ID for future single sign-ons.
14. RBTickets creates a login session for the user, and redirects them to the protected portal page they were originally trying to access.

Example 9-1 SAML request from RBTickets to RBTelco to exchange artifact for assertion

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header></soapenv:Header>
  <soapenv:Body>
    <samlp:Request xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
      IssueInstant="2005-08-23T10:17:51Z" MajorVersion="1" MinorVersion="1"
      RequestID="FIMREQ_e2d802ac-0105-e7f6-078e-9ab00c5a78ff">
      <ds:Signature Id="uuide2d802b4-0105-ee2c-b579-9ab00c5a78ff">
        <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="#"FIMREQ_e2d802ac-0105-e7f6-078e-9ab00c5a78ff">
            <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="samlp
ds"></xc14n:InclusiveNamespaces>
        </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"></ds:DigestMethod>

<ds:DigestValue>Qg18H1Q8d1mA2UB/wBODTLAIXgM=</ds:DigestValue>
        </ds:Reference>
</ds:SignedInfo>

<ds:SignatureValue>F6qIsFq116aKEACKyrqVcofSFp2VposhtDazC0htNcmE8P5WB22JY9wkwR5+
afCuHnSLECEsb0cGZ+acPe1/xZEgjqTQR0IUgmegegbrSCrk6IeqmkE0+1E1XR1qVtz71xa01a1LL/w
jersHDVjQjibq0acrDTSbF4Z0e0g4qYU=</ds:SignatureValue>
        <ds:KeyInfo>
        <ds:X509Data>

<ds:X509Certificate>MIICrzCCAhigAwIBAgIBDjANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQQDEXN
maW0ucmVhYy5pYm0uY29tMQswCQYDVQQGEwJVVzEMMAoGA1UEChMDSUJNMB4XDTA1MDYwMTIxNT
IzNFoXDTAwMDYxNjIxNTIzNFowSzEoMCYGA1UEAxQfcmJ0aWNRZXRzX3JidGVsY28ucmJ0aWNRZXRzL
mNvbTElMAkGA1UEBhMCVVMxEjAQBgNVBAoTCVJCVG1ja2V0czCBnzANBgkqhkiG9w0BAQEFAA0BJQAw
gYkCgYEAresQnHEXNckehLvV/Mocq8TvAiJMgA8P+VPa1IicGCRcY7ENON7dzK+B8FV5XSd+tF6vpXJ
kELs1kYtiKUPer4q1cF6ehfvLNJuMm4m+Qx7F9eStMJN1RvtXB7jKbe8UtzuQIOeAXcKJu9uSYW95V2
pbo61GsgdQKjBv+bZB558CAwEAAa0BtDCBsTAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBbQVedA2vUX7I
Eo4I/iVYYOR9ttpZjBhBgNVHSMEWjBYgBRA0z474m+/yN+mhtBtCGxeD8uRMAE9pDsw0TEcMBoGA1UE
AxMTZmltLnJlZGJvb2suaWJtLmNvbTElMAkGA1UEBhMCVVMxDDAKBgNVBAoTA01CTYIBATALBgNVHQ8
EBAMCBLAwEgYJYIZIAYb4QgENBAUWA2hwaDANBgkqhkiG9w0BAQQFAA0BgQAz1/a4QKeZFN39oVbm3u
CWMXD8ZYnde4/2iWD2PFmgZw7QJPWwouLJ+VfXEi34s39skMFe1AxjBPJ0X1QqZK8SrmHi1BJyGDCH
xzqgk5S/DqHun7bJaDSgFgzqEqkg8oCNGEp0pM8ABn0GHI44utpDZ8A4w0q6odLJXvG27kg0A==</ds
:X509Certificate>
        </ds:X509Data>
        </ds:KeyInfo>
</ds:Signature>

<samlp:AssertionArtifact>AA0/pac7rYy13xjECRyohKvb5qdv02BbrDK0s9kewMJK9RGyx9TMJW
PG</samlp:AssertionArtifact>
        </samlp:Request>
</soapenv:Body>
</soapenv:Envelope>

```

Example 9-2 SAML P response containing Liberty assertion

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header />
  <soapenv:Body>
    <samlp:Response xmlns:ds="http://www.w3.org/2000/09/xmlsig#"
xmlns:lib="urn:liberty:iff:2003-08"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"

```



```

xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
InResponseTo="FIMREQ_e2d802ac-0105-e7f6-078e-9ab00c5a78ff"
IssueInstant="2005-08-23T10:18:02Z" MajorVersion="1" MinorVersion="1"
ResponseID="FIMRSP_e2d82dcd-0105-e8ec-1663-f750dd5f48d2">
  <samlp:Status>
    <samlp:StatusCode Value="samlp:Success" />
  </samlp:Status>
  <saml:Assertion
AssertionID="Assertion-uuide2d81651-0105-f1b0-5b09-f750dd5f48d2"
IssueInstant="2005-08-23T10:17:56Z"
Issuer="https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty" MajorVersion="1"
MinorVersion="2" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:lib="urn:liberty:iff:2003-08"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="lib:AssertionType">
    <saml:Conditions NotBefore="2005-08-23T10:16:56Z"
NotOnOrAfter="2005-08-23T10:19:56Z">
      <saml:AudienceRestrictionCondition>
        <saml:Audience>https://www.rbtickets.com/ITFIM/sps/liberty12/liberty</saml:Audi
ence>
      </saml:AudienceRestrictionCondition>
    </saml:Conditions>
    <saml:AuthenticationStatement
AuthenticationInstant="2005-08-23T10:17:56Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
SessionIndex="uuide2d7f943-0105-e597-6dc5-f750dd5f48d2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="lib:AuthenticationStatementType">
      <saml:Subject
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="lib:SubjectType">
        <saml:NameIdentifier
Format="urn:liberty:iff:nameid:federated"
NameQualifier="https://www.rbtickets.com/ITFIM/sps/liberty12/liberty">uuide2d81
61f-0105-f7a8-212a-f750dd5f48d2</saml:NameIdentifier>
        <saml:SubjectConfirmation>
          <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:artifact</saml:Confirma
tionMethod>
        </saml:SubjectConfirmation>
      </saml:Subject>
    </saml:AuthenticationStatement>
    <ds:Signature Id="uuide2d81654-0105-f4e9-76bb-f750dd5f48d2"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>

```

```
    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <ds:Reference
URI="#Assertion-uuide2d81651-0105-f1b0-5b09-f750dd5f48d2">
    <ds:Transforms>
    <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
    <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 xsi
lib" />
    </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <ds:DigestValue>0XmpfIiJDRhNKzhMZMJhWQHnIuw=</ds:DigestValue>
    </ds:Reference>
</ds:SignedInfo>

<ds:SignatureValue>EJdbnjQxFk6yZTquCUJ8jPZ+07bq04nUou1cbbvQejIAxodWSUVCFsyw7swX
9mPE5ik9aU2h9c34mFDcxKk47AhS9ST4jt5rJ1Awq4J+u9HTRkbztkJkMfAGDpf17Sdy6nFG4uaobkB
1jw1xDUeRFRrEPmGsvZct/jazEzivyuoM=</ds:SignatureValue>
    <ds:KeyInfo>
    <ds:X509Data>

<ds:X509Certificate>MIICqzCAhSgAwIBAgIBCTANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQDExN
maW0ucmVhYm9vay5pYm0uY29tMQswCQYDVQGEwJVUzEMMAoGA1UEChMDSUJNMB4XDTA1MDYwMTIxMz
cxN1oXDEwMDYxNjIxMzcwN1owRzEmMCQGA1UEAxQdcmluZWRzbnR1bG9vMIGfMAOGCSqGSY3DQEBAAUA4GNADCB1QKB
gQDmXXWJHs2RxFksCE3p+QpnyHBoPcAa/ph0Dvfuof4FsjccfLRQLJhURrY1j8u26YwsI0bcCuZft1b
Rb0UnUwVcIzPdGu1nNn/P21hy/SY7qSm/v1d6FVg5nP7ouEvjsFUT3wgoas+ww2JJDjUGo951tf+z1
WYKX063R1jP10BfQIDAQABo4G0MIGxMAwGA1UdEwEB/wQCMAAwHQYDVRR0BBYEFETwMgToY0cx7LRS3
Tz0oojf1Gh5MGEGA1UdIwRaMFjAFEDTPjvib7/I36aG0G0IbF4Py5ExoT2k0zA5MRwwGgYDVQDExNm
aW0ucmVhYm9vay5pYm0uY29tMQswCQYDVQGEwJVUzEMMAoGA1UEChMDSUJNggEBMA5GA1UdDwQEAwI
EsDASBg1ghkgBhvCAQOEBrYDaHBoMAOGCSqGSY3DQEBBAUA4GBAGSpG3tCj1DQGs/RU7WkOGA1AP
30dsap9pYGZ/6sQ6bg1SIFsIJfhi fwAscGMuAL33vqCkFuXh6hgwttjRgPLIFy00qUiNXd0EmPOH3q/
7L0KXWITmLQ9h211xKz3fI7bXW11PirEptGdkyrPzZ4smxvs10DnytK6KnHzIwg9jEF</ds:X509Cer
tificate>

    </ds:X509Data>
    </ds:KeyInfo>
</ds:Signature>
</saml:Assertion>
</samlp:Response>
</soapenv:Body>
```

9.4.2 Single sign-on to partners (Liberty)

After the name identifier has been established, the user now has the ability to perform single sign-on between RBTelco and the partners. The interaction is the same between RBTelco and each of the partners, so for illustration we just show the interaction with RBTickets in Figure 9-11. A detailed description of the interaction follows the figure.

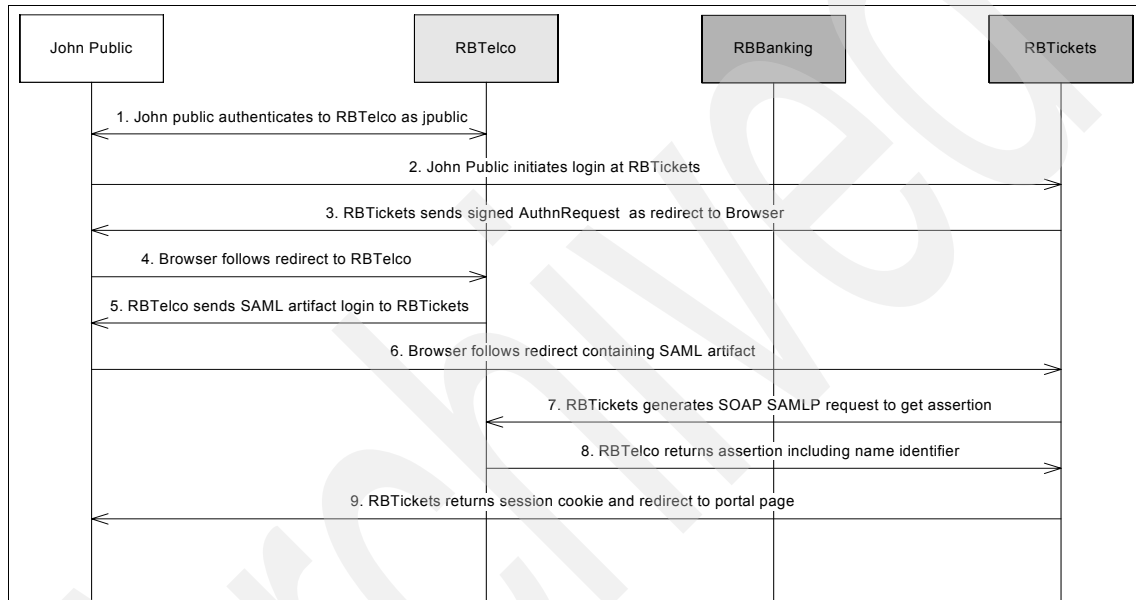


Figure 9-11 Liberty Browser/Artifact Profile single sign-on flow

Below is a detailed description of the interaction shown in Figure 9-11:

1. John Public authenticates to RBTelco.
2. John initiates a single sign-on. This is always initiated at the service provider, and the link looks like:

<https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/login?RelayState=https://www.rbtickets.com>

3. RBTickets sends a signed authentication request to RBTelco as a redirect to the browser. This redirect looks like:

https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/login?RequestID=FIMREQ_e333f8d7-0105-ed13-058f-9ab00c5a78ff&MajorVersion=1&MinorVersion=2&IssueInstant=2005-08-23T11%3A58%3A18Z&ProviderID=https%3A%2F%2Fwww.rbtickets.com%2

```
FITFIM%2Fsps%2Fliberty12%2Fliberty&IsPassive=false&ProtocolProfile=http%3A%2F%2Fprojectliberty.org%2Fprofiles%2Fbrws-art&RelayState=uuid333f889-0105-fff0-3343-9ab00c5a78ff&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1&Signature=BVNhmPpoAW9mDMkZ%2B5wtm%2F1FB9oEa3bbHdMnWifz0GVygUu%2Bg91GhGN0hKJJ1EUvmbmdhqjyctgA%0D%0Aqc4YhwHWzAwHpFFCPs15nHgT%2F1yUABTIzQyQqiYXGE30Lp4c0C9y9pK1Jjf9gBdGQ1tbFpCWJ1s%0D%0AeRx8qrepwoD1c7XfMA0%3D
```

4. The browser follows redirect to RBTelco.
5. RBTelco generates an assertion for the user, and redirects with a SAML artifact to RBTickets. This redirect looks like:

```
https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/login?RelayState=uid333f889-0105-fff0-3343-9ab00c5a78ff&SAMLart=AA0%2Fpac7rYy13xjECRyohKvb5qdv05oyZEmSoxhumqGoiim38sSyXiBT
```
6. The browser follows redirect to RBTickets.
7. RBTickets generates a SOAP SAML request to RBTelco to exchange the artifact for an assertion. This follows precisely the same format as that shown in Example 9-1 on page 257.
8. RBTelco responds with the assertion, as per Example 9-2 on page 258.
9. RBTickets generates a session for the user and sends back to the browser a session cookie along with a redirect to the originally requested resource defined by the RelayState in step 2 (in our case the RBTickets portal page).

9.4.3 Single sign-off

This interaction diagram picks up after John Public has authenticated to RBTelco, and performed single sign-on to both RBBanking and RBTickets. The single logout used in this scenario follows the HTTP redirect profile and is shown in Figure 9-12 on page 263. A detailed description of the interaction follows the figure. There are also two other types of logout profiles supported by liberty, called HTTP Get and SOAP/HTTP. Any of these could have been used for our scenario, and our choice was arbitrary.

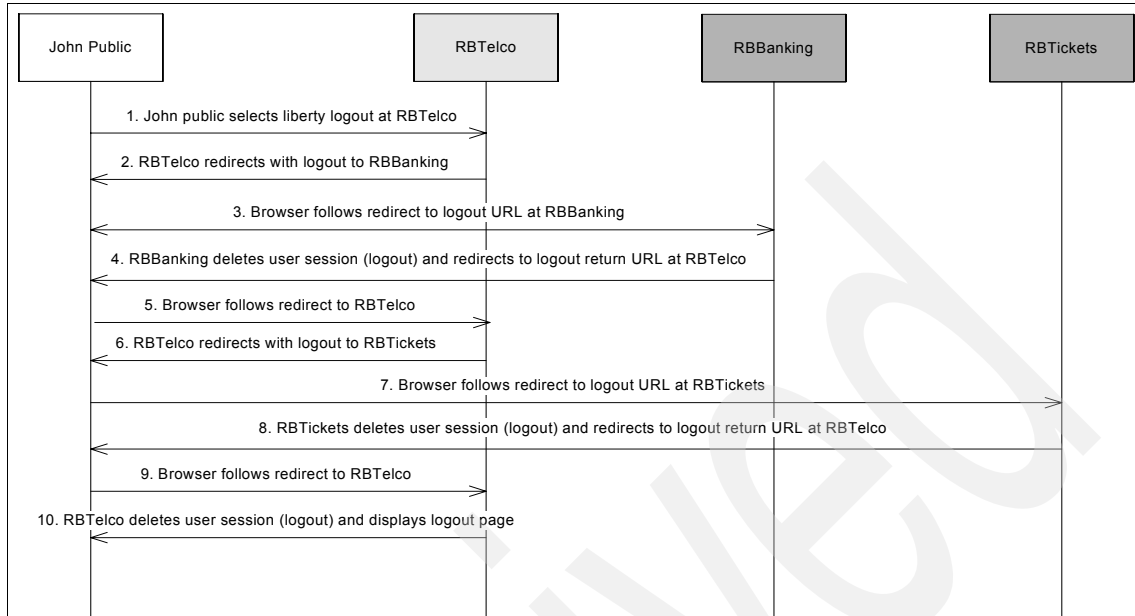


Figure 9-12 Liberty HTTP redirect single logout flow

A detailed description of the interaction shown in Figure 9-12 follow:

1. John Public selects the Liberty logout link at RBTelco. This looks like:

<https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/slo>

2. RBTelco redirects with a logout URL for RBBanking. The redirect is to this URL:

https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/slo?RequestID=FIMREQ_e32cbb14-0105-f86e-81d1-f750dd5f48d2&MajorVersion=1&MinorVersion=2&IssueInstant=2005-08-23T11%3A50%3A23Z&ProviderID=https%3A%2F%2Fwww.rbtelco.com%2FITFIM%2F2Fsp%2Flibertyfed%2Fliberty&NameQualifier=https%3A%2F%2Fwww.rbbanking.com%2FITFIM%2F2Fsp%2Fliberty12%2Fliberty&NameFormat=urn%3A1liberty%3A1iff%3Anameid%3Afederated&NameIdentifier=uuide1a7278a-0105-f66b-b71f-f750dd5f48d2&SessionIndex=uuide32bd21c-0105-fd1a-21a0-f750dd5f48d2&RelayState=uuide32bd21c-0105-fd1a-21a0-f750dd5f48d2&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1&Signature=sgtXUexW1T%2BYhotH3KrZEdG8UggCcVpa4vgvli09siMF1Ms1RBb0QpB4w7Mjh50ImH0k0SP97CB%0D%0AmBw8U2vNdEK2y2xKZKd%2FZL60dRyw7qxFeq61Br6ksqEGpHycSgXGFQT7xCB6A4UIEuM5W%2BIPuqo5%0D%0A7gBuEeiu0M%2F7CrbQZz8%3D

3. The browser follows redirect to RBBanking.
4. RBBanking deletes the user's session, and redirects back to the RBTelco logout return URL. This redirect looks like:

https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/sloreturn?ResponseID=FIMRSP_e32c6f21-0105-e064-3809-93e6c6e30d05&InResponseTo=FIMREQ_e32cbb14-010

5-f86e-81d1-f750dd5f48d2&MajorVersion=1&MinorVersion=2&IssueInstant=2005-08-23T11%3A50%3A04Z&Recipient=https%3A%2F%2Fwww.rbtelco.com%2FITFIM%2Fsp%2Flibertyfed%2Fliberty&ProviderID=https%3A%2F%2Fwww.rbanking.com%2FITFIM%2Fsp%2Fliberty12%2Fliberty&Value=samlp%3ASuccess&RelayState=uuide32bd21c-0105-fd1a-21a0-f750dd5f48d2&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1&Signature=AJpISya7Qv19rS6mwAFN69wErMcqdTyHVFX0FRDDVpynU5HwsdotaZDmxLKMs3Ynj1R9vrfX3DGD%0D%0AIjVcvzheDM22j8Tyg%2Fp3rR76EM9mXhCcW38qqa01pv5ATqGfVmogLKIVQVn21%2FZThQ2yyUKDU%2BUi%0D%0AgMiSm9xVtowsf74iR2U%3D

5. The browser follows the logout return URL to RBTelco.
6. RBTelco redirects with a logout URL for RBTickets. The redirect is to this URL:

https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/slo?RequestID=FIMREQ_e32cbb29-0105-f162-fc44-f750dd5f48d2&MajorVersion=1&MinorVersion=2&IssueInstant=2005-08-23T11%3A50%3A23Z&ProviderID=https%3A%2F%2Fwww.rbtelco.com%2FITFIM%2Fsp%2Flibertyfed%2Fliberty&NameQualifier=https%3A%2F%2Fwww.rbtickets.com%2FITFIM%2Fsp%2Fliberty12%2Fliberty&NameFormat=urn%3A1iberty%3Aiff%3Anameid%3Afederated&NameIdentifier=uuide2d8161f-0105-f7a8-212a-f750dd5f48d2&SessionIndex=uuide32bd21c-0105-fd1a-21a0-f750dd5f48d2&RelayState=uuide32bd21c-0105-fd1a-21a0-f750dd5f48d2&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1&Signature=P20Jn6bYGXXi39fadpYebnJNuyxsk5T%2FLK0BF7NAUdKZ8x1KvtZYvRTDb%2BdIIII6XGsw50FwBf0%0D%0AyN3ZdJeHkNLz%2BeNgYK6U%2BzW0u2PB%2FmSgLbGn870r17r78NZ%2Bws1fTTCjyj0kfy0dN7%2FfZsDBZQBp%0D%0A97Jps4aY1H%2F1sUVv94o%3D

7. The browser follows redirect to RBTickets.
8. RBTickets deletes the user's session, and redirects back to the RBTelco logout return URL. This redirect looks like:

https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/sloreturn?ResponseID=FIMRSP_e32c9617-0105-ec75-4a38-9ab00c5a78ff&InResponseTo=FIMREQ_e32cbb29-0105-f162-fc44-f750dd5f48d2&MajorVersion=1&MinorVersion=2&IssueInstant=2005-08-23T11%3A50%3A14Z&Recipient=https%3A%2F%2Fwww.rbtelco.com%2FITFIM%2Fsp%2Flibertyfed%2Fliberty&ProviderID=https%3A%2F%2Fwww.rbtickets.com%2FITFIM%2Fsp%2Fliberty12%2Fliberty&Value=samlp%3ASuccess&RelayState=uuide32bd21c-0105-fd1a-21a0-f750dd5f48d2&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1&Signature=JbywY2G9I4xARNi8NayWzC%2FDGfZ91gu6t%2BKTDCG1uSroAc1zk6h39UKEZqa50F3EWnkyXPN4WATt%0D%0Agp6zNNxf06AEfwN0bQk1ce7AuVIBRB%2BCTA3KX0tYY0LgC0ySATQVcv1reSWB0QIZ2Ub%2F%2B2WM2Rs%0D%0AXjEiOVmMwBdvJOB08%3D

9. The browser follows redirect to RBTelco.
10. RBTelco deletes user session information and sends the browser the logout success page.

9.5 Configuration data

The following references will assist with the installation and configuration of Tivoli Federated Identity Manager:

- ▶ *IBM Tivoli Identity Manager Installation Guide Version 6.0*, GC32-1667-00, discusses the installation of Tivoli Federated Identity Manager.
- ▶ *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00, contains basic information about configuring the Tivoli Federated Identity Manager runtime and information on configuring federations.
- ▶ Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, contains information about configuring Tivoli Federated Identity Manager for use with WebSEAL.

9.5.1 Identity provider configuration at RBTelco

Configuring Liberty federation at RBTelco consists of the following tasks:

- ▶ Importing RBTelco signing keys
- ▶ Configuring Tivoli Federated Identity Manager for Liberty as an identity provider
- ▶ Configuring the service provider partner for RBTickets and RBBanking
- ▶ Configuring Access Manager policy for the federation URLs

Importing RBTelco keys

Appendix C, “Keys and certificates” on page 425, contains information about the key strategy used for all use cases. In particular, note that for this federation configuration the `rbtelco-signing.jks` key file was imported into Tivoli Federated Identity Manager. This contains the signing key used to sign the Liberty Messages sent to the partners. Also, the `rbtelco-partners.jks` keffiyeh was imported and contains the partners public keys to verify their signed Liberty messages.

Configuring RBTelco as a Liberty identity provider

Detailed information about configuring an identity provider to use Liberty is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00. This section discusses the specific configuration parameters used for RBTelco.

Federation Properties

*Federation Name
libertyfed

My Role: Identity Provider

*Company Name
RBTelco

Company URL
[]

Contact Person

First Name [] Last Name []

Email Address []

Phone Number []

Liberty v1.2 Token Module Configuration

*Amount of time the assertion is valid after being issued (seconds)
120

[View/Edit Identity Mapping Rule...](#)

My Single Sign-On Properties

SSO Protocol: Liberty ID-FF 1.2

Common Data

*Provider ID
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty

Succinct Provider ID: v6WnO62Mpd8YxAkcqISr2+anbzs=

SOAP Endpoint
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/soap

Single Sign-On

*Single Sign-On Service URL
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/login

Liberty Message Lifetime (in seconds)
120

Liberty Artifact Lifetime (in seconds)
120

Require Consent to Federate

Figure 9-13 RBTelco Liberty Federation configuration part 1

Liberty Enabled Client/Proxy (LECP)

LECP Providers (enter as comma delimited list)

Register Name Identifier (RNI)

RNI Service URL

RNI Return URL

HTTP Redirect
 SOAP/HTTP

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET
 HTTP Redirect
 SOAP/HTTP

Identity Provider Introduction (IPI)

Common DNS Domain

Common Domain Hostname

Digital Signature

Sign Liberty Messages

Signing Key Identifier

Figure 9-14 RBTelco Liberty Federation configuration part 2

Figure 9-13 on page 266 and Figure 9-14 show all the information needed to configure the Liberty federation for RBTelco. For more information about the identity mapping including the complete XSLT mapping see “RBTickets mapping for use case 3” on page 416.

Configuring service provider partners for RBTelco

The screenshot shows a web-based configuration interface for a service provider partner. The window title is "Partner Properties". The configuration is organized into several sections:

- *Company Name:** RBTickets
- Company URL:** https://www.rbtickets.com
- Contact Person:** Fields for First Name, Last Name, Email Address, and Phone Number.
- Member of Federation:** libertyfed
- Partner Role:** Service Provider
- Status:** Enabled (with a Disable button)
- Liberty v1.2 Token Module Configuration:** Includes a field for attribute types (set to *) and a "View/Edit Identity Mapping Rule..." button.
- Single Sign-On Properties:**
 - Common Data:**
 - *Provider ID:** https://www.rbtickets.com/ITFIM/sps/liberty12/liberty
 - Succinct Provider ID:** WYVlc19XAqj5AE4IQnIt7cSeqo=
 - SOAP Endpoint:** https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/soap
 - Single Sign-On:**
 - *Assertion Consumer Service URL:** https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/login
 - Register Name Identifier (RNI):**
 - RNI Service URL:** https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/rni
 - RNI Return URL:** https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/rnireturn
 - Authentication Method:** Radio buttons for HTTP Redirect (selected) and SOAP/HTTP.

Figure 9-15 RBTelco Liberty Federation partner 1 (RBTickets) configuration part 1

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET or HTTP Redirect
 SOAP/HTTP

Digital Signature

Require Partner to Sign Liberty Messages
Verification Key Identifier

SOAP SSL Connection Parameters (used ONLY if SOAP Endpoint is https)

Select Server Validation Certificate

Partner Requires Client Basic Authentication
Client Basic Authentication Username

Client Basic Authentication Password

Partner Requires Client Certificate Authentication
Select Client Certificate

Figure 9-16 RBTelco Liberty Federation partner 1 (RBTickets) configuration part 2

Figure 9-15 on page 268 and Figure 9-16 show all the information needed to configure the Liberty federation partner RBTickets with RBTelco. The identity mapping rule is already defined in the federation and is therefore left empty in the partner configuration.

Partner Properties

*Company Name
RBBanking

Company URL
https://www.rbbanking.com

Contact Person

First Name Last Name

Email Address

Phone Number

Member of Federation: libertyfed
Partner Role: Service Provider
Status: Enabled

Liberty v1.2 Token Module Configuration

Include the following attribute types (a "*" means include all types)

Single Sign-On Properties

Common Data

*Provider ID
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty
Succinct Provider ID: wQ/tcStjIcwIfNerWwEzflNeEUo=
SOAP Endpoint
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/soap

Single Sign-On

*Assertion Consumer Service URL
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/login

Register Name Identifier (RNI)

RNI Service URL
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/rni
RNI Return URL
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/rnireturn

HTTP Redirect
 SOAP/HTTP

Figure 9-17 RBTelco Liberty Federation partner 2 (RBBanking) configuration part 1

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET or HTTP Redirect
 SOAP/HTTP

Digital Signature

Require Partner to Sign Liberty Messages

Verification Key Identifier

SOAP SSL Connection Parameters (used ONLY if SOAP Endpoint is https)

Select Server Validation Certificate

Partner Requires Client Basic Authentication
 Client Basic Authentication Username

Client Basic Authentication Password

Partner Requires Client Certificate Authentication
 Select Client Certificate

Figure 9-18 RBTelco Liberty Federation partner 2 (RBBanking) configuration part 2

Figure 9-17 on page 270 and Figure 9-18 show all the information needed to configure the Liberty federation partner RBBanking with RBTelco. The identity mapping rule is already defined in the federation and is therefore left empty in the partner configuration.

Configuring Access Manager policy at RBTelco

There are several Liberty endpoints exposed at RBTelco. These appear in the object space as:

```
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/auth
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/ftn
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/ftninitial
```

```
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/ftnreturn  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/login  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/rni  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/rniinitial  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/rnireturn  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/slo  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/sloreturn  
/WebSEAL/<webseal-server-object>/ITFIM/sps/libertyfed/liberty/soap
```

The policy for Liberty at RBTelco follows these logical rules:

- ▶ RBTelco requires all customers to use SSL. An Access Manager protected object policy will be used to enforce this.
- ▶ The Liberty URLs should be accessible to any retail customer, but not to business partner users that have single signed on to RBTelco from BigCorp. An Access Manager access control list will enforce this.
- ▶ The Liberty *soap* URL receives signed requests without certificate authentication, so unauthenticated access should be allowed to this endpoint.
- ▶ The *auth* URL is used for reauthentication in the case of a special Liberty sign-on flag called ForceAuthn. For that purpose, an Access Manager Protected Object Policy (POP) which forces reauthentication will be attached to the auth URL.

Example 9-3 shows the `pdadmin` commands used to create and apply the aforementioned policy.

Example 9-3 Use case 3 Access Manager policy for RBTelco

```
pop create rbtelco_ssl  
pop modify rbtelco_ssl set qop privacy  
pop attach /WebSEAL/<webseal_server>/ITFIM rbtelco_ssl  
  
acl create rbtelco_retail  
acl modify rbtelco_retail set group iv-admin TcmdsbsvaBRrx1  
acl modify rbtelco_retail set group webseal-servers Tgmdbsrx1  
acl modify rbtelco_retail set user sec_master TcmdsbsvaBRrx1  
acl modify rbtelco_retail set any-other Trx  
acl modify rbtelco_retail set user bigcorp_guest T  
acl modify rbtelco_retail set unauthenticated T  
acl attach /WebSEAL/<webseal_server>/ITFIM/sps/libertyfed rbtelco_retail  
  
acl create rbtelco_unauth  
acl modify rbtelco_unauth set group iv-admin TcmdsbsvaBRrx1  
acl modify rbtelco_unauth set group webseal-servers Tgmdbsrx1  
acl modify rbtelco_unauth set user sec_master TcmdsbsvaBRrx1  
acl modify rbtelco_unauth set any-other Trx  
acl modify rbtelco_unauth set unauthenticated Trx
```

```
acl attach /WebSEAL/<webseal_server>/ITFIM/sps/libertyfed/liberty/soap
rbtelco_unauth

pop create rbtelco_reauth
pop modify rbtelco_reauth set attribute reauth true
pop attach /WebSEAL/<webseal_server>/ITFIM/sps/libertyfed/liberty/auth
rbtelco_reauth
```

9.5.2 RBTickets service provider configuration data

Configuring the Liberty federation at RBTickets consists of the following tasks:

- ▶ Importing RBTickets signing keys
- ▶ Configuring Tivoli Federated Identity Manager for Liberty as a service provider
- ▶ Configuring the identity provider partner
- ▶ Configuring Access Manager policy for the federation URLs

Importing RBTickets keys

Appendix C, “Keys and certificates” on page 425, contains information about the key strategy used for all use cases. In particular, note that for this federation configuration the `rbtickets-signing.jks` keffiyeh was imported into Tivoli Federated Identity Manager. This contains the signing key used to sign the Liberty Messages. Also, the `rbtickets-partners.jks` keffiyeh was imported and contains the partner’s public key to verify the signed Liberty Messages.

Configuring RBTickets as a Liberty service provider

Detailed information about configuring a service provider to use Liberty is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00. This section discusses the specific configuration parameters used for RBTickets.

Federation Properties

*Federation Name
liberty12

My Role: Service Provider

*Company Name
RBTickets

Company URL
https://www.rbtickets.com

Contact Person

First Name Last Name

Email Address

Phone Number

[View/Edit Identity Mapping Rule...](#)

My Single Sign-On Properties

SSO Protocol: Liberty ID-FF 1.2

Common Data

*Provider ID
https://www.rbtickets.com/ITFIM/sps/liberty12/liberty
Succinct Provider ID: WYVlc19XAqj5AE4IQriIt7cSeqo=
SOAP Endpoint
https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/soap

Single Sign-On

*Assertion Consumer Service URL
https://www.rbtickets.com/ITFIM/sps/liberty12/liberty/login

Liberty Message Lifetime (in seconds)

Single Sign-On is Passive (Identity Provider does not interact with user)

Force Identity Provider to authenticate user

Use Browser Artifact Profile for Single Sign-On

Figure 9-19 RBTickets Liberty Federation configuration part 1

Liberty Enabled Client/Proxy (LECP)

LECP Providers (enter as comma delimited list)

Register Name Identifier (RNI)

RNI Service URL

RNI Return URL

HTTP Redirect
 SOAP/HTTP

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET or HTTP Redirect
 SOAP/HTTP

Digital Signature

Sign Liberty Messages

Signing Key Identifier

Figure 9-20 RBTickets Liberty Federation configuration part 2

Figure 9-19 on page 274 and Figure 9-20 show all the information needed to configure the Liberty federation for RBTickets. For more information about the identity mapping including the complete XSLT mapping see “RBTickets mapping for use case 3” on page 416.

Configuring an identity provider partner for RBTickets

Partner Properties

*Company Name
RBTelco

Company URL

Contact Person

First Name Last Name

Email Address

Phone Number

Member of Federation: liberty12
Partner Role: Identity Provider

Status: Enabled

Liberty v1.2 Token Module Configuration

Username to be used for anonymous users

Single Sign-On Properties

Common Data

*Provider ID
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty

Succinct Provider ID: v6WnO62Mpd8YxAkcqISr2+anbzs=

SOAP Endpoint
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/soap

Single Sign-On

*Single Sign-On Service URL
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/login

Register Name Identifier (RNI)

RNI Service URL
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/rni

RNI Return URL
https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/rnireturn

HTTP Redirect
 SOAP/HTTP

Figure 9-21 RBTickets Liberty Federation partner configuration part 1

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET
 HTTP Redirect
 SOAP/HTTP

Digital Signature

Require Partner to Sign Liberty Messages

Verification Key Identifier

SOAP SSL Connection Parameters (used ONLY if SOAP Endpoint is https)

Select Server Validation Certificate

Partner Requires Client Basic Authentication

Client Basic Authentication Username

Client Basic Authentication Password

Partner Requires Client Certificate Authentication

Select Client Certificate

Figure 9-22 RBTickets Liberty Federation partner configuration part 2

Figure 9-21 on page 276 and Figure 9-22 show all the information needed to configure the Liberty federation partner RBTelco with RBTickets. The identity mapping rule is already defined in the federation and is therefore left empty in the partner configuration.

Configuring Access Manager policy at RBTickets

There are several Liberty endpoints exposed at each of the Liberty partners. These appear in the object space as:

```
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/ftn
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/ftninitial
```

```
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/ftnreturn  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/login  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/rni  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/rniinitial  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/rnireturn  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/slo  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/sloreturn  
/WebSEAL/<webseal-server-object>/ITFIM/sps/liberty12/liberty/soap
```

The policy for Liberty at each of the Liberty partners follows these logical rules:

- ▶ All customers are required to use SSL. An Access Manager protected object policy will be used to enforce this.
- ▶ The Liberty *login* URL must be configured to allow unauthenticated access so that users can log in. The Liberty *soap* URL receives signed requests without certificate authentication, so unauthenticated access should be allowed to this endpoint also. An Access Manager ACL will be used for this.
- ▶ All other Liberty URLs can be accessed by any authenticated user. We will let default-webseal inherited ACL policy take care of this.

Example 9-4 shows the `pdadmin` commands used to create and apply the aforementioned policy.

Example 9-4 Use case 3 Access Manager policy for Liberty partners

```
pop create rbpartner_ssl  
pop modify rbpartner_ssl set qop privacy  
pop attach /WebSEAL/<webseal_server>/ITFIM rbpartner_ssl  
  
acl create rbpartner_unauth  
acl modify rbpartner_unauth set group iv-admin TcmdbsvaBRrx1  
acl modify rbpartner_unauth set group webseal-servers Tgmdbsrx1  
acl modify rbpartner_unauth set user sec_master TcmdbsvaBRrx1  
acl modify rbpartner_unauth set any-other Trx  
acl modify rbpartner_unauth set unauthenticated Trx  
acl attach /WebSEAL/<webseal_server>/ITFIM/sps/liberty12/liberty/login  
rbpartner_unauth  
acl attach /WebSEAL/<webseal_server>/ITFIM/sps/liberty12/liberty/soap  
rbpartner_unauth
```

9.5.3 RBBanking service provider configuration data

Configuring the Liberty federation at RBBanking consists of the following tasks:

- ▶ Importing RBBanking signing keys
- ▶ Configuring Tivoli Federated Identity Manager for Liberty as a service provider

- ▶ Configuring the identity provider partner
- ▶ Configuring Access Manager policy for the federation URLs

Importing RBBanking keys

Appendix C, “Keys and certificates” on page 425, contains information about the key strategy used for all use cases. In particular, note that for this federation configuration the `rbbanking-signing.jks` keffiyeh was imported into Tivoli Federated Identity Manager. This contains the signing key used to sign the Liberty Messages. Also, the `rbbanking-partners.jks` keffiyeh was imported and contains the partner’s public key to verify the signed Liberty messages.

Configuring RBBanking as a Liberty service provider

Detailed information about configuring a service provider to use Liberty is available in the *IBM Tivoli Identity Manager Administration Guide Version 6.0*, GC32-1668-00. This section discusses the specific configuration parameters used for RBBanking.

Federation Properties

*Federation Name
liberty12

My Role: Service Provider

*Company Name
RBBanking

Company URL
https://www.rbbanking.com

Contact Person

First Name Last Name

Email Address

Phone Number

[View/Edit Identity Mapping Rule...](#)

My Single Sign-On Properties

SSO Protocol: Liberty ID-FF 1.2

Common Data

*Provider ID
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty
Succinct Provider ID: wQ/tcStjIcwIfNerWwEzflNeEUo=
SOAP Endpoint
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/soap

Single Sign-On

*Assertion Consumer Service URL
https://www.rbbanking.com/ITFIM/sps/liberty12/liberty/login

Liberty Message Lifetime (in seconds)

Single Sign-On is Passive (Identity Provider does not interact with user)

Force Identity Provider to authenticate user

Use Browser Artifact Profile for Single Sign-On

Liberty Enabled Client/Proxy (LECP)

LECP Providers (enter as comma delimited list)

Figure 9-23 RBBanking Liberty Federation configuration part 1

Register Name Identifier (RNI)

RNI Service URL

RNI Return URL

HTTP Redirect
 SOAP/HTTP

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET or HTTP Redirect
 SOAP/HTTP

Digital Signature

Sign Liberty Messages

Signing Key Identifier

Figure 9-24 RBBanking Liberty Federation configuration part 2

Figure 9-23 on page 280 and Figure 9-24 show all the information needed to configure the Liberty federation for RBBanking. For more information about the identity mapping including the complete XSLT mapping see “RBBanking mapping for use case 3” on page 415.

Configuring an identity provider partner for RBBanking

Partner Properties

*Company Name
RBTelco

Company URL

Contact Person

First Name Last Name

Email Address

Phone Number

Member of Federation: liberty12
Partner Role: Identity Provider
Status: Enabled

Liberty v1.2 Token Module Configuration

Username to be used for anonymous users
not-used

Single Sign-On Properties

Common Data

*Provider ID
<https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty>

Succinct Provider ID: v6WnO62Mpd8YxAkcqISr2+anbzs=
SOAP Endpoint
<https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/soap>

Single Sign-On

*Single Sign-On Service URL
<https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/login>

Register Name Identifier (RNI)

RNI Service URL
<https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/rni>

RNI Return URL
<https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty/rnireturn>

HTTP Redirect
 SOAP/HTTP

Figure 9-25 RBBanking Liberty Federation partner configuration part 1

Federation Termination Notification (FTN)

FTN Service URL

FTN Return URL

HTTP Redirect
 SOAP/HTTP

Single Logout (SLO)

SLO Service URL

SLO Return URL

HTTP GET
 HTTP Redirect
 SOAP/HTTP

Digital Signature

Require Partner to Sign Liberty Messages
Verification Key Identifier

SOAP SSL Connection Parameters (used ONLY if SOAP Endpoint is https)

Select Server Validation Certificate

Partner Requires Client Basic Authentication
Client Basic Authentication Username

Client Basic Authentication Password

Partner Requires Client Certificate Authentication
Select Client Certificate

Figure 9-26 RBBanking Liberty Federation partner configuration part 2

Figure 9-25 on page 282 and Figure 9-26 show all the information needed to configure the Liberty federation partner RBTelco with RBBanking. The identity mapping rule is already defined in the federation and is therefore left empty in the partner configuration.

Configuring Access Manager policy at RBBanking

The policy for RBBanking is identical to the policy for RBTickets. Please see “Configuring Access Manager policy at RBTickets” on page 277 for details.

9.6 Assumptions/implementation notes

This section contains use case 3 specific information that may be of interest to the reader.

9.6.1 InfoService integration

During lab development for this use case we made use of a Tivoli Federated Identity Manager API called the InfoService. At the time of writing, it is Tivoli's stated intention to release this API to customers in the very near future. This API is noteworthy because it provides a way to determine federation membership information for a particular user.

The InfoService APIs allow you to query both the federation membership information and the URL endpoints of Liberty federations and partners. This allows you to dynamically build the links for:

- ▶ Federation of accounts
- ▶ Register name identifier
- ▶ Federation termination
- ▶ Single sign-on
- ▶ Single sign-off

Utilizing this API, we are able to generate customized, meaningful portal pages for individual users based on which partners they are federated with, and which partners they are not federated with. For example, when user jpublic was federated with the RBBanking partner, but not with RBTickets, his portal page at RBTelco looks like that shown in Figure 9-27 on page 285. Then after federating with RBTickets, it looks like that shown in Figure 9-28 on page 286. This was made possible by utilizing the InfoService APIs from the portal JSP, and querying for information about jpublic and the federation named liberty.

Similarly, on the service provider (for example, RBTickets), you can utilize the InfoService APIs during operations like reauthentication to determine the list of partner identity providers that a user has federations with. This allows you to customize the login page at the service provider, or in the case of a single Identity provider partner immediately redirect to the identity provider for login.



Figure 9-27 Portal page for jpublic federated with only RBBanking



Figure 9-28 Portal page for jpublic federated with both RBBanking and RBTickets

9.6.2 Page customizations

One of the more detailed tasks when deploying a Liberty use case is to get the application look and feel working correctly both for federated partners and for integration with the Access Manager point of contact server (WebSEAL in our case). This section discusses the WebSEAL pages that were customized for the look and feel we generated at both the identity provider and the service

providers. We chose a couple of common look and feel scenarios, though many more are possible.

RBTelco

The primary customization done at RBTelco was the portal page. This utilized the Tivoli Federated Identity Manager InfoService, as discussed previously, to generate a custom look and feel for the user based on which partners he was federated with.

The login page at RBTelco was not modified beyond adding some graphics to brand it as RBTelco. Although it is possible for users to single sign-on to RBTelco from BigCorp (see use case 2), we did not consider this an option we wanted to expose from the RBTelco login page. The only way we wished to expose this to BigCorp employees is via the BigCorp portal page, as a push-style login.

The only other customization at RBTelco was the logout page. We added javascript to delete all WebSphere-generated session cookies from the browser, and we detected BigCorp users so that we did not display the bigcorp_guest user ID during the logout. Example 9-5 shows the contents of the logout.html page from RBTelco.

Example 9-5 logout.html at RBTelco

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<!-- Copyright (C) 2000 Tivoli Systems, Inc. -->
<!-- Copyright (C) 1999 IBM Corporation -->
<!-- Copyright (C) 1998 Dascom, Inc. -->
<!-- All Rights Reserved. -->
<HTML>
<BODY>
<H1>Logout Successful</H1>
<SCRIPT TYPE="text/javascript">
    // delete WebSphere session cookies
    document.cookie = 'AMWEBJCT!%2Fapps!JSESSIONID=0; expires=Fri, 13-Apr-1970
00:00:00 GMT';
    document.cookie = 'AMWEBJCT!%2Fapps!LtpaToken=0; expires=Fri, 13-Apr-1970
00:00:00 GMT';
    document.cookie = 'AMWEBJCT!%2Fapps!LtpaToken2=0; expires=Fri, 13-Apr-1970
00:00:00 GMT';
    document.cookie = 'AMWEBJCT!%2FITFIM!JSESSIONID=0; expires=Fri, 13-Apr-1970
00:00:00 GMT';
</SCRIPT>
<SCRIPT TYPE="text/javascript">
    var username = "%USERNAME%";
    if ( username == "bigcorp_guest" ) {
        document.write( "RBTelco thanks you for your business.<BR>" );
    }
</SCRIPT>
```

```

        document.write( "<A HREF=\"https://www.bigcorp.com\">Return to
BigCorp</A>" );
    }
    else {
        document.write( username + " has logged out.<BR>" );
    }
</SCRIPT>
</BODY>
</HTML>

```

Partners (RBTickets, RBBanking)

The main customization at the partners is that the WebSEAL login page was modified to be generated by a jsp. Example 9-6 shows the WebSEAL login.html.

Example 9-6 WebSEAL login.html at partners

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<!-- Copyright (C) 2000 Tivoli Systems, Inc. -->
<!-- Copyright (C) 1999 IBM Corporation -->
<!-- Copyright (C) 1998 Dascom, Inc. -->
<!-- All Rights Reserved. -->
<HTML>
<FORM METHOD=POST ACTION="/apps/RBTickets/unprotected/generateLogin.jsp">
<INPUT TYPE="HIDDEN" NAME="USERNAME" VALUE="%USERNAME%">
<INPUT TYPE="HIDDEN" NAME="ERROR" VALUE="%ERROR%">
<INPUT TYPE="HIDDEN" NAME="URL" VALUE="%HTTPS_BASE%URL%">
</FORM>
<SCRIPT TYPE="text/javascript">
    setTimeout('document.forms[0].submit()', 0);
</SCRIPT>
</BODY>
</HTML>

```

There are a couple of things you could do in the generateLogin.jsp:

- ▶ Provide links or redirect directly to the single sign-on URL. What you do depends upon the particular circumstances of your deployment and your desired user experience. For example, if you only have one service provider federation configured, and do not wish to support local login (this requires pre-populating the Liberty name identifiers for all users), then it is quite practical to redirect immediately to the single sign-on URL, which will in turn complete the WAYF process (if more than one identity provider partner), and then generate a sign-on request to the identity provider partner. In our lab we just prompt for user name/password login, and do not expose the fact that we have identity provider partners from the partner login page. This was by choice of application design. We expect customers to only perform federated single sign-on from the provided links in the RBTelco portal page.

- ▶ Detect if this is a reauthentication (due, for example, to WebSEAL session expiry). You can tell if it is a reauthentication if the %USERNAME% macro is populated with a non-empty value. In this case you can use the InfoService APIs to determine whether this user has any federated identity provider partners, and automatically provide him with a list of partners with which he may reauthenticate, or automatically redirect if there is only one. This is the scenario we tested in the lab (prompting with a list). We tuned the RBTickets WebSEAL to have a short session expiry, and when reauthentication was required we detected the user name and queried the InfoService to determine the identity provider partners for which this user was federated. The sign-in URL is also returned from the InfoService. The resulting generated login page during reauthentication is shown in Figure 9-29.

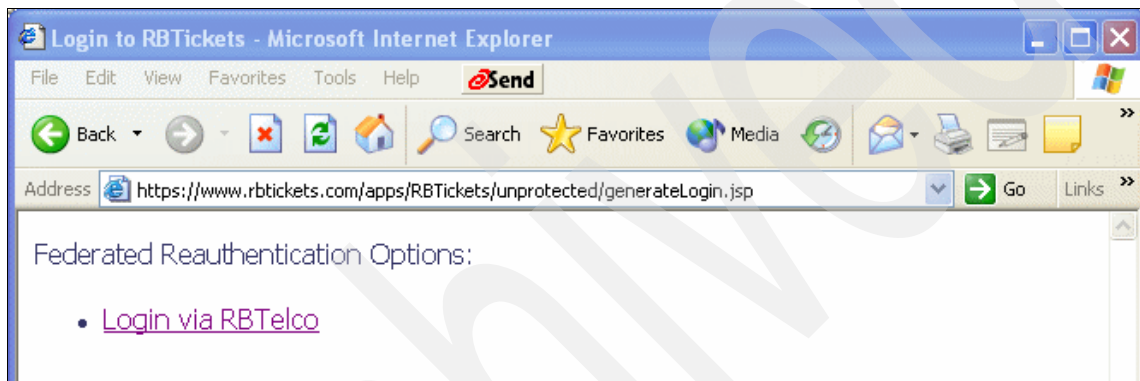


Figure 9-29 RBTickets login page for johnp during reauthentication

Archived

Use case 4 - Web services security management

In this use case we show several ways that the Web services security management components of Tivoli Federated Identity Manager can extend the WS-Security functionality available within WebSphere. We demonstrate how Web services security management can be used internally within an enterprise to pass client identity and attribute information between an application running on WebSphere Application Server and a WebSphere Web Services Gateway. We then show how Web services security management can be used on the outbound side of a WebSphere Web Services Gateway to add a SAML assertion as a security token in a Web services request, allowing that request to be honored by a federated Web service hosted at another company. Finally, we show how such a security token would be processed by the company that hosts the federated Web Service including the verification, user ID, and attribute mapping, authorization, and token transformation that is associated with being a security token consumer. Throughout we highlight any significant differences between Web services security management and the federated single sign-on capabilities of Tivoli Federated Identity Manager.

To get the most out of this use case, the reader should be familiar with the Tivoli Federated Identity Manager Web services security management Guide as well as the prerequisite publications that it assumes including the Web services specifications for WS-Security ("Web Services Security (WS-Security)")

specification” on page 470) and WS-Trust (“Web Services Trust Language (WS-Trust) specification” on page 470).

Archived

10.1 Scenario details

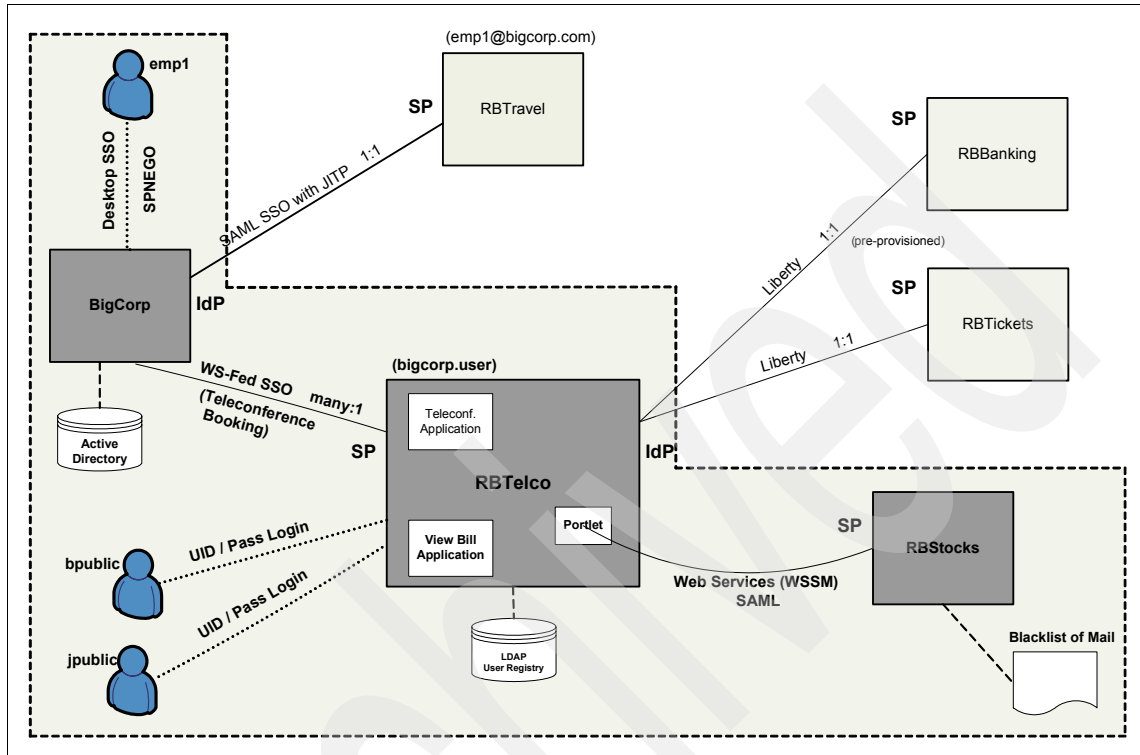


Figure 10-1 Use case 4 logical architecture

The components and actors that are present in this use case are highlighted by the grey box in the lower left corner of the diagram shown in Figure 10-1, “Use case 4 logical architecture” on page 293. Here we focus on how RBTelco generates a secure Web service request to RBStocks on behalf of the clients that have authenticated to their system. The diagram shows that the initial user authentication can be handled directly by RBTelco in the case where their retail customers are authenticating with user ID and password, or it can result from a federated single sign-on with one of their business partner customers such as BigCorp via WS-Federation.

10.1.1 Contract

RBStocks has agreed to expose to RBTelco a Web service that provides stock quotes on the condition that:

- ▶ RBTelco will employ WS-Security to ensure the integrity and confidentiality of the Web services request and that the request will contain a signed SAML 1.1 assertion as the security token.
- ▶ The SAML assertion itself will contain two extended attributes.
 - A user_home attribute will identify where the client originally authenticated.
 - An email_address attribute will contain the e-mail address of the client.
- ▶ RBStocks will provide realtime stock quotes to RBTelco's corporate customers and delayed stock quotes to RBTelco's retail customers (those identified with a user_home attribute of RBTelco). RBStocks at its discretion can blacklist any client, based on the e-mail address.

Example 10-1 shows an example of an SAML 1.1 assertion as would be used by a retail customer of RBTelco.

Example 10-1 SAML 1.1 assertion for Web services request from RBTelco to RBStocks

```
<saml:Assertion xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="Assertion-uuid2d3419c9-0105-ec42-9011-85f6225bfc32"
IssueInstant="2005-07-19T03:47:33Z" Issuer="https://www.rbtelco.com/rbstocks"
MajorVersion="1" MinorVersion="1">
  <saml:Conditions NotBefore="2005-07-19T03:46:33Z"
NotOnOrAfter="2005-07-19T03:57:33Z">
    <saml:AudienceRestrictionCondition>

      <saml:Audience>urn:itfim-wssm:wsgwsoaphttp1:soaphttpengine:WSGW_BUS:StockQuotes
ervice:wsgw_server1_SOAPHTTPChannel1_InboundPort</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement AuthenticationInstant="2005-07-19T03:47:33Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">jpublic@rbtelco
.com</saml:NameIdentifier>
    </saml:Subject>
  </saml:AuthenticationStatement>
  <saml:AttributeStatement>
    <saml:Subject>
```

```

        <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">jpublic@rbtelco
.com</saml:NameIdentifier>
        </saml:Subject>
        <saml:Attribute AttributeName="user_home"
AttributeNamespace="http://rbtelco.com/user_home">
        <saml:AttributeValue>RBTelco</saml:AttributeValue>
        </saml:Attribute>
    </saml:AttributeStatement>
    <ds:Signature Id="uuid2d341c82-0105-fb29-b4e7-85f6225bfc32"
xmlns:ds="http://www.w3.org/2000/09/xmlsig#">
    <ds:SignedInfo>
        <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#rsa-sha1" />
        <ds:Reference
URI="#Assertion-uuid2d341c9-0105-ec42-9011-85f6225bfc32">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-signature" />
                <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" />
                </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#sha1" />
            <ds:DigestValue>5MIL7k2wS04ZqCFUzDzTgC+d/7o=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>

    <ds:SignatureValue>cILho2HYwtm893vypPUWbHHGq+KpocIxo+q7J30z8K0qvvrKvAjP+w819bDA
F77Ux4IppvDCCE9t6AV00421xmh9yCjWBSR4pNx883KSBvR8MKa3zNeSALONMigURKnYBuaX4NHnNuev
ycgRQinm7/8Cx+DR1viviG3dh375VISVE=</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>

            <ds:X509Certificate>MIICqjCCAhOgAwIBAgIBCjANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQQDEXN
maW0ucmVkaW9yYm9yY29tMQswCQYDVQQGEwJVUzEMMAoGA1UEChMDSUNNMB4XDTE1MDYwMTIxMz
gyNV0xODTEwMDYxNjIxMzgyNVowRjE1MCMGA1UEAxQccmJ0ZWxjb19yYnN0b2Nrcy5yYnR1bGNvLmNvb
TElMAkGA1UEBhMCMVVMxEDA0BgNVBAoTB1JCVGVsY28wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGB
AL+2o0OriISrXrrs1f1sCa/9j4a9VAnBup/pix37esCWLv7i4qjUKof6JJ+QgtJQDGZ8Q0bcmRm9KR
t5pR2GSA NiIaBExLnThmW0Zf18L9epKYDn/kD7Aw5P1UhJaPdG7aSi+SF+5PriQT420t1oD9JD9duWB
qP5dc2pdxG3hZAgMBAAGjgbQwgbEwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUDBihhccfoFuBoSAzj
8c+SLXohj4wYQYDVROjBFowWIAUQNM+0+Jvv8jfpobQbQhsXg/LkTGHpaQ7MDkxHDAaBgNVBAMTE2Zp
bS5yZWRib29rLmli bS5jb20xCzAJBgNVBAYTA1VTMwQwYDVQQKEwNJKoZlCAQEWcWYDVROPBQAQAgS
wMBIGCWGSAGG+EIBDQFFgNocGgWdQYJKoZIhvcNAQEEBQADgYEAYBRKR1f709wcLsPbfn7962BJw1

```

```
U29txs046oFUzPUyBRaZK0fwwMN2yZZz06nUpBmViX0mzofsv+KvmXAN1f0BQa8zX3F41tPspg8t1Vo
GEq79uUWj7s+/B5GDjgh92NU6WIOGjiOcaI31nwIckM38RcBqHPgnHbAiQ0uoxfoss=</ds:X509Cer
tificate>
  </ds:X509Data>
  </ds:KeyInfo>
  </ds:Signature>
</saml:Assertion>
```

10.1.2 User experience

Unlike the previously described use cases that were focused on Federated Single Sign-on, there is little new of interest here in the way of user interaction. In this use case we show how Tivoli Federated Identity Manager adds value to WS-Security, and this occurs in the context of a server-to-server interaction. The only true client interaction is with an RBTelco-hosted JSP that will make the Web services request to RBStocks on the client's behalf. Nonetheless, below we show the client experience based on the three possible responses from the Stock Quote Web service:

- ▶ Stock Quote Web service response to RBTelco corporate customer with realtime access

BigCorp employees are automatically authenticated to the BigCorp portal due to the SPNEGO authentication achieved via WebSEAL and Windows desktop single sign-on. Figure 10-2 on page 297 shows the portal page at BigCorp that Employee One is presented with.

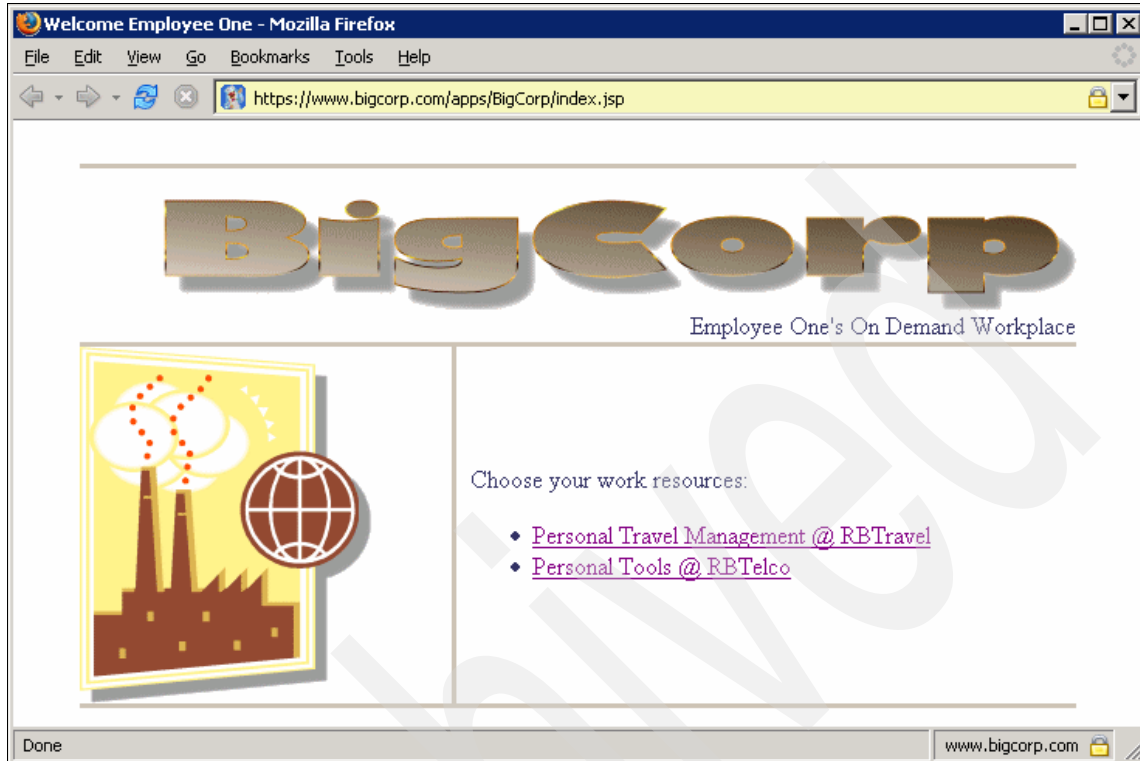


Figure 10-2 Employee One's view of the BigCorp portal page

Figure 10-3 on page 298 shows the portal page seen after clicking the link for Personal Tools @ RBTelco and being authenticated at RBTelco via WS-Federation, as described in Chapter 8, "Use case 2 - WS-Federation" on page 219.



Figure 10-3 Employee emp1's view of the RBTelco portal page

Figure 10-4 on page 299 shows the page seen by Employee One with realtime access after clicking the Get a Stock Quote and providing the symbol of the company for which he wants the quote.

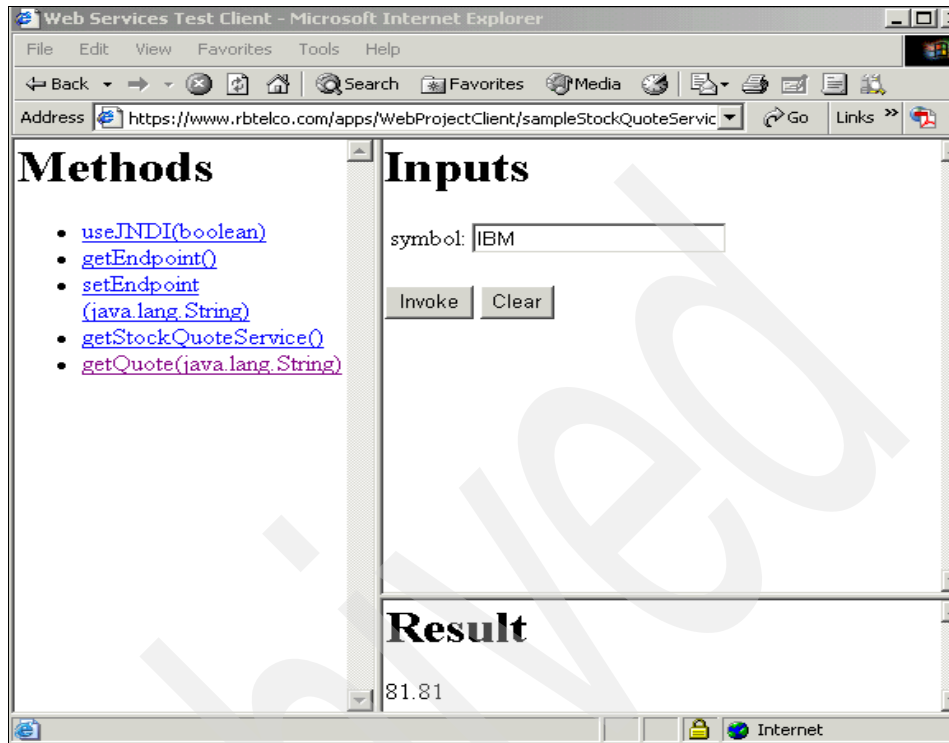


Figure 10-4 Employee emp1's view of the Stock Quote service

- ▶ Stock Quote Web service response to RBTelco retail customer with delayed access

Figure 10-5 on page 300 shows the portal page that retail customer jpublic will see after he successfully authenticates with a user name and password to RBTelco.



Figure 10-5 Retail customer jpublic's view to the RBTelco portal page

Figure 10-6 on page 301 shows the screen that retail customer jpublic with delayed access will see when he clicks the Get a Stock Quote link and then provides the symbol for which he wants the quote.

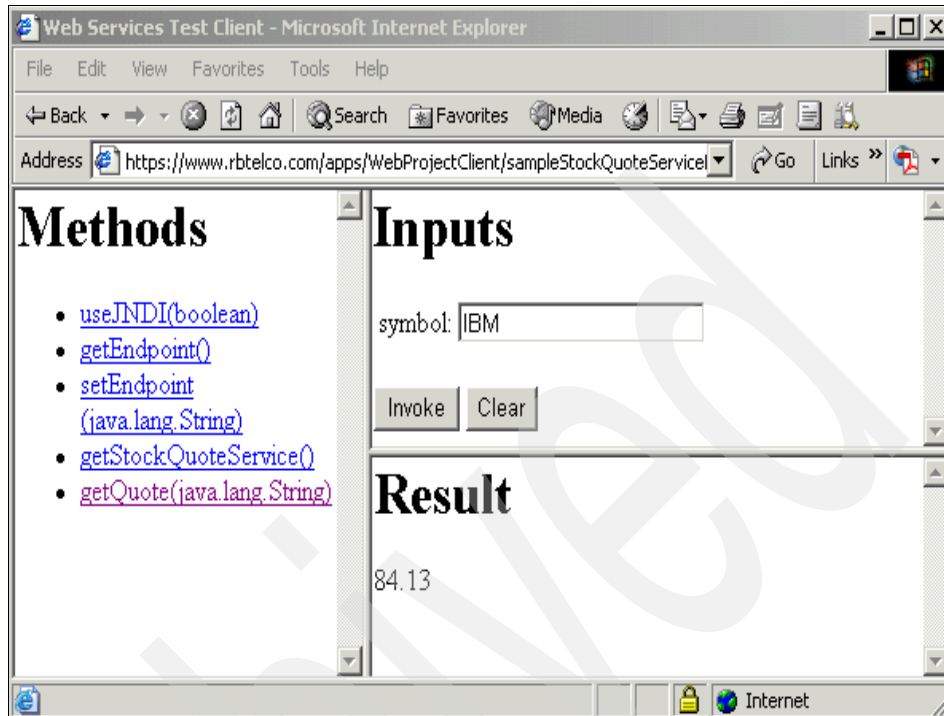


Figure 10-6 Retail customer jpublic's view of the Stock Quote service

- ▶ Stock Quote Web service response to a blacklisted client

Regardless of whether the client has realtime or delayed access, if they have been blacklisted they will end up seeing the same screen as above, but after selecting a symbol will get a message indicating that a runtime exception was thrown by the security handler instead of seeing a numeric result. An administrator looking at the security token service logs on RBStocks would see that the blacklist had been checked and this client was contained in the list.

10.2 Functionality

RBTelco and RBStocks each make use of Web Service Security Management to extend the functionality of WS-Security support provided by WebSphere. In this scenario WS-Security is employed to achieve message integrity and confidentiality of the Web service request that RBTelco sends to RBStocks. However, all of the security token processing shown in this scenario is handled by Web services security management. While WebSphere will natively support

several security token types such as the Username token and the X509 Credential token, the richer token types such as the SAML security token and the Tivoli Access Manager binary security token that are used in this scenario are only available with the Web services security management component of Tivoli Federated Identity Manager. In addition to the richer set of security tokens, Web services security management provides the ability to interface with Tivoli Access Manager to perform authorization checking for a Web service prior to the service being invoked, and it provides the ability to perform identity mapping of asserted IDs. In this scenario all of these Web services security management functions are employed.

10.2.1 Web services security management at RBTelco

RBTelco makes use of Web services security management in two ways. Web services security management is used internally by RBTelco to pass an authenticated user's identity and extended attributes in an Access Manager binary security token from a WebSphere application server hosting their Web service clients to its Web Services Gateway. RBTelco makes further use of Web services security management when it transforms the Access Manager binary security token into an SAML assertion and adds it to the Web services request on the outbound side of the Web Services Gateway. The SAML assertion is used for authentication and authorization at RBStocks prior to invoking the Web service.

10.2.2 Web services security management at RBStocks

RBStocks, unlike RBTelco, has opted not to use a WebSphere Web Services Gateway. RBStocks simply interfaces with Web services security management from their WebSphere application server hosting the Stock Quote Web service. They employ Web Service Security Management to validate the SAML assertions coming from their customers, to map the identity passed in the assertion into a local user, to check whether the given user is authorized to the service, and finally to transform the received assertion into a format acceptable to their WebSphere JAAS login configuration so that a security context for the user can be created in WebSphere.

10.3 Partners involved

The corporations involved in this use case are RBTelco and RBStocks.

10.3.1 RBTelco

RBTelco, among its other customer offerings described in the previous use cases, provides access to the Stock Quote Web service hosted by RBStocks.

10.3.2 RBStocks

RBStocks hosts the Stock Quote Web service, which can supply either 15-minute delayed quotes or real-time quotes. RBStocks maintains a custom blacklist, which is checked during client authentication to prohibit access to anyone that has been barred from use of the service.

10.4 Interaction description

Figure 10-7 on page 304 depicts the interaction between a WebSEAL server, the WebSphere Application Server hosting the Web service client, and the WebSphere Web Service Gateway, all located at RBTelco with the WebSphere Application Server at RBStocks, which hosts the Stock Quote Web service.

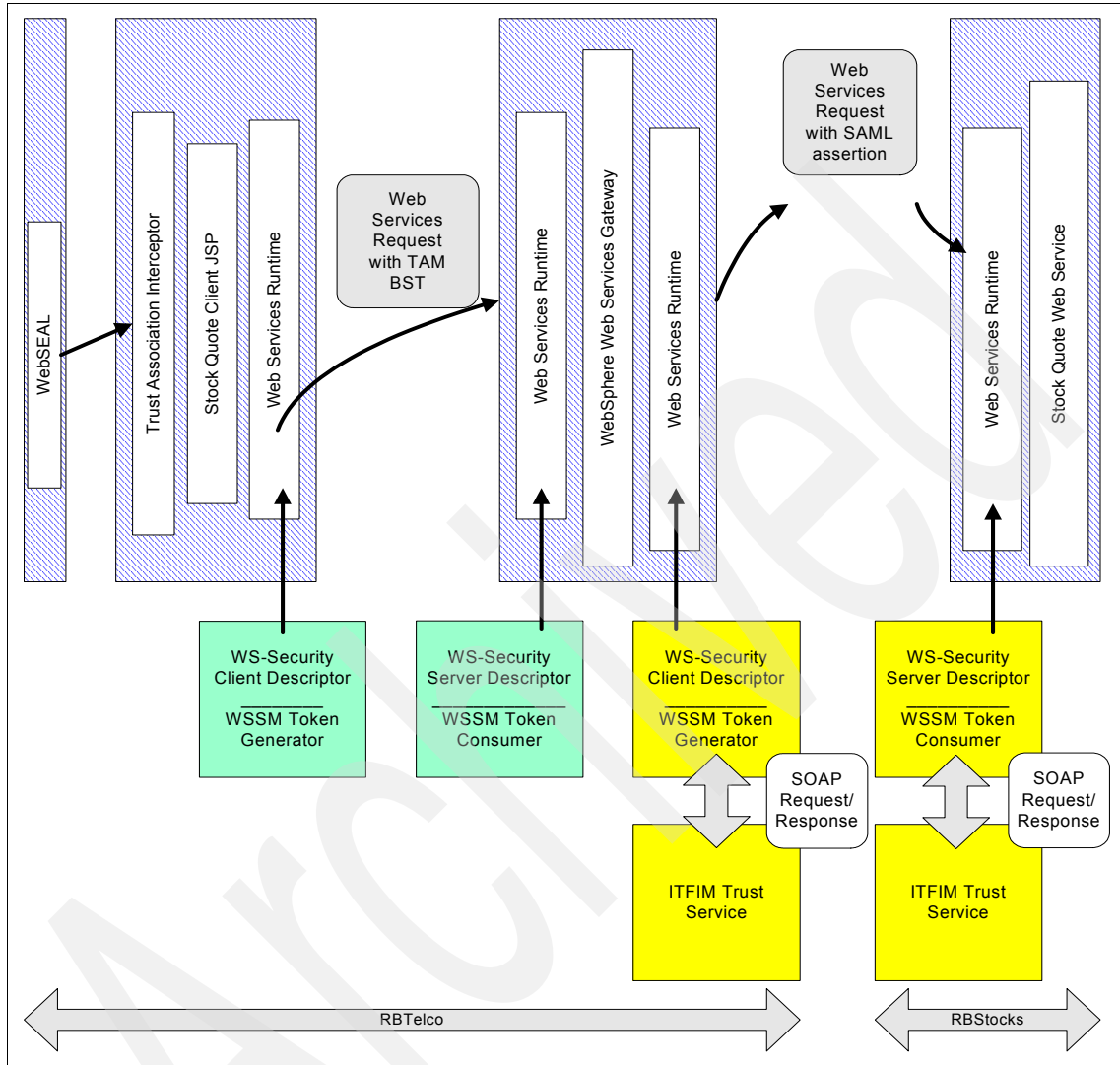


Figure 10-7 Security token processing as the Web Service request traverses the servers

There are four points of particular interest in the above diagram, each of which will be discussed in turn in 10.4.1, “Web services security management Token Generator with Access Manager binary security token callback handler” on page 305, through 10.4.4, “Web services security management Token Consumer with SAML Assertion login module” on page 315. Briefly, these are:

- ▶ The use of a Web services security management Token Generator on the application server at RBTelco to create an Access Manager binary security

token based on the authenticated user's Access Manager credentials. The Access Manager binary security token is then inserted into the Web service request as it passes to the gateway.

- ▶ The use of a Web services security management Token Consumer on the inbound side of the gateway at RBTelco, which processes the request using a JAAS login configuration that understands the Access Manager binary security token and can create a security context for the identified user on the gateway.
- ▶ The use of a Web services security management Token Generator that interfaces with the Tivoli Federated Identity Manager trust service on the outbound side of the gateway at RBTelco to create a signed SAML assertion. The SAML assertion is inserted as the security token in the Web service request before the gateway forwards it to RBStocks.
- ▶ The use of a Web services security management Token Consumer that interfaces with the Tivoli Federated Identity Manager trust service when the Web service request is received at RBStocks. Its primary functions are to:
 - Validate the signature on the signed assertion.
 - Perform blacklist checking based on the e-mail address.
 - Perform identity mapping (for realtime or delayed quotes) based on the user_home attribute.
 - Transform the signed SAML assertion, which was received into an unsigned SAML assertion for which there is a JAAS login configuration to create a security context for the user.

10.4.1 Web services security management Token Generator with Access Manager binary security token callback handler

Web services security management is employed at the WebSphere application server at RBTelco to generate an Access Manager binary security token that contains the identity of the client as derived from the client's Access Manager credential. The Access Manager credential itself was created when the client authenticated to WebSEAL and the Trust Association Interceptor on the application server caused a JAAS login to occur before the client accessed the protected JSP. The Access Manager binary security token created by Web services security management is inserted as the security token in the Web service request as it is passed to the gateway. The specific steps required to configure the creation of this token type are covered in 10.5, "Configuration data" on page 319. For now we simply illustrate in Example 10-2 on page 306 the end result of the token generation in order to clarify the interaction occurring between the RBTelco WebSphere application server hosting the Stock Quote Web service client and the RBTelco WebSphere Web Services Gateway.

Example 10-2 Stock Quote client Web service request with an Access Manager binary security token in header

```
<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wss:BinarySecurityToken EncodingType="http://ibm.com/2004/01/itfim/base64encode"
ValueType="http://ibm.com/2004/01/itfim/ivcred"
wsu:Id="uudf704b527-0104-e6f7-1043-e02ceabe9775"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        BAKs3DCCArcMADCCArEwggKtAgIFEDAzMC8wHgIE1Z6ytgIDA0NeAgIR2QICAJ8CARgEBgAMKRJhDAwNYmlnY29ycF9ndWV
        zdDAAAgEBMIICbTCCAmkwLAWSQ...RkJPVnpGdWJUUR1Q3N5ZVZSeFRHTnpkM0pWYkVsb1RpMWhaalZPWnp0cmFHWmhkRX
        BvZVhwSFp3PT0EAA==</wss:BinarySecurityToken>
      </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
      <p680:getQuote xmlns:p680="http://StockQuote">
        <symbol>IBM</symbol>
      </p680:getQuote>
    </soapenv:Body>
  </soapenv:Envelope>
```

Note: There are a number of security tokens that may take the BASE64 encoded binary security token format. They are distinguished by value type. We can tell that this is an Access Manager binary security token because in the above example we see:

```
ValueType="http://ibm.com/2004/01/itfim/ivcred"
```

In Example 10-1 on page 294 it can be seen from the document that WS-Security has not been employed to either sign or encrypt the Web service request between the WebSphere Application Server and the gateway. This is simply a decision that was made by RBTelco to rely on transport level security when communicating between their internal servers. WS-Security certainly could have been used as well to achieve transport-independent integrity and confidentiality. We will see how WS-Security is used for signing and encryption when we look at the outbound request from the RBTelco's WebSphere Web Services Gateway to RBStocks.

10.4.2 Web services security management Token Consumer with Access Manager Credential login module

When the request reaches the gateway at RBTelco, Web services security management is employed once again. The Token consumer is configured to require an Access Manager binary security token and perform a JAAS login for the client using the token. Once again, details of the configuration required to achieve this are covered in 10.5, "Configuration data" on page 319.

10.4.3 Web services security management Token Generator with Web services security management Callback handler

At the outbound side of the gateway Web services security management is configured to interface with the Tivoli Federated Identity Manager trust service. Web services security management provides the Access Manager credential of the logged in user and requests that a signed SAML assertion be returned. An example of this request is shown in Example 10-3.

Example 10-3 SOAP request to the trust service to generate a signed SAML assertion from an Access Manager credential

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <wst:RequestSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
      <wst:Issuer xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wsa:Address>http://www.rbtelco.com/internal</wsa:Address>
      </wst:Issuer>
      <wsp:AppliesTo xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference>
          <wsa:Address>urn:itfim-wssm:wsgwsoaphttp1:soaphttpengine:WSGW_BUS:StockQuoteService:wsgw_server
1_SOAPHTTPChannel1_InboundPort</wsa:Address>
          </wsa:EndpointReference>
        </wsp:AppliesTo>
      <wst:Base>
        <wss:BinarySecurityToken EncodingType="http://ibm.com/2004/01/itfim/base64encode"
ValueType="http://ibm.com/2004/01/itfim/ivcred"
wsu:Id="uuidf704b527-0104-e6f7-1043-e02ceabe9775"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
BAKs3DCCArcMADCCArEwg...BdOFBQUE4QT10Q0dKQ2FtVnpTRkJpVnpGdWJUUR1Q3N5ZVZSeFRHTnpkMOpWYkVsb1RpM
Whaa1ZPwPncmFhWmhkRXBvZVhwSFp3PT0EAA==</wss:BinarySecurityToken>
      </wst:Base>
    </wst:RequestSecurityToken>
  </soapenv:Body>
</soapenv:Envelope>
```

```

    </wst:Base>

<wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Validate</wst:RequestType>
  </wst:RequestSecurityToken>
</soapenv:Body>
</soapenv:Envelope>

```

Example 10-4 shows how the trust service would respond to the request in Example 10-3 on page 307 by providing a signed SAML assertion that can be inserted as the security token of the Web service request to the Stock Quote Web service.

Example 10-4 Trust service response providing a signed SAML Assertion for the given client

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsa:Action
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Validate</wsa:Action>
    </soapenv:Header>
    <soapenv:Body>
      <wst:RequestSecurityTokenResponse wsu:Id="uuidf704f98f-0104-e4d2-0bd9-a0e298abb70b"
        xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wst:RequestedSecurityToken>
          <saml:Assertion AssertionID="Assertion-uuidf704f951-0104-f735-d1bb-a0e298abb70b"
            IssueInstant="2005-07-08T15:16:35Z" Issuer="https://www.rbtelco.com/rbstocks" MajorVersion="1"
            MinorVersion="1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
            <saml:Conditions NotBefore="2005-07-08T15:15:35Z"
              NotOnOrAfter="2005-07-08T15:26:35Z">
              <saml:AudienceRestrictionCondition>
                <saml:Audience>urn:itfim-wssm:wsgwsoaphttp1:soaphttpengine:WSGW_BUS:StockQuoteService:wsgw_server1_SOAPHTTPChannel1_InboundPort</saml:Audience>
              </saml:AudienceRestrictionCondition>
            </saml:Conditions>
            <saml:AuthenticationStatement AuthenticationInstant="2005-07-08T15:16:35Z"
              AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
              <saml:Subject>
                <saml:NameIdentifier
                  Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">emp1@bigcorp.com</saml:NameIdentifier>
              </saml:Subject>
            </saml:AuthenticationStatement>
          </wst:RequestedSecurityToken>
        </wst:RequestSecurityTokenResponse>
      </soapenv:Body>
    </soapenv:Envelope>

```

```

    <saml:AttributeStatement>
      <saml:Subject>
        <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">emp1@bigcorp.com</saml:NameIdentifier>
      </saml:Subject>
      <saml:Attribute AttributeName="user_home"
AttributeNamespace="http://rbtelco.com/user_home">
        <saml:AttributeValue>BigCorp</saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
    <ds:Signature Id="uuidf704f958-0104-feba-c2ac-a0e298abb70b"
xmlns:ds="http://www.w3.org/2000/09/xmlsig#">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#rsa-sha1" />
        <ds:Reference URI="#Assertion-uuidf704f951-0104-f735-d1bb-a0e298abb70b">
          <ds:Transforms>
            <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-signature" />
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <xc14n:InclusiveNamespaces PrefixList="saml ds"
xmlns:xc14n="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1" />
          <ds:DigestValue>gN6wzAgjvqlTdIZGdKsf0IKC/po=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>

      <ds:SignatureValue>u0Q3qU6DMiXRnv/9eecVIeIz1rfiwgHm03kR4DWDt/nWuHXjLgmb/hnq2driSHpy8Af0bLw9kHrx
y0wqPp0Yh/UBHFhf47ZeKY5Wnkc0vCdwAh3RxrXBu/ssy9xveqbxGcgqplzDwmufxkNxSvBFvQifQBy1wJQvmhAh0GD8neg
=</ds:SignatureValue>
      <ds:KeyInfo>
        <ds:X509Data>

        <ds:X509Certificate>MIICqjCCAhOgAwIBAgIBCjANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQQQExNmaW0ucmVhYm9vY29tMQswCQ...Zz06nUpBmViX0mzofsv+KvmXAN1f0BQa8zX3F41tPspg8t1VoGEq79uUWj7s+/B5GDjgh92NU6WIO
GjiOcaI31nwIckM38RcBqHPgnHbAiQ0uoxfoss=</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
</saml:Assertion>
</wst:RequestedSecurityToken>
<wst:RequestedAttachedReference
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <wss:SecurityTokenReference>

```

```

        <wss:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">A
sassertion-uuidf704f951-0104-f735-d1bb-a0e298abb70b</wss:KeyIdentifier>
        </wss:SecurityTokenReference>
    </wst:RequestedAttachedReference>
    <wst:Status>
        <wst:Code>http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid</wst:Code>
    </wst:Status>
</wst:RequestSecurityTokenResponse>
</soapenv:Body>
</soapenv:Envelope>

```

Example 10-5 shows the complete Web service request including the signed SAML 1.1 assertion as it would be presented to the Stock Quote Web service if WS-Security was not being employed for signing and encryption of the SAML token and message body. The next example shows the same request with WS-Security signing and encryption in effect.

Example 10-5 Web service request to the Stock Quote Web service with signed SAML assertion

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://StockQuote">
    <env:Header>
        <wsse:Security env:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <saml:Assertion AssertionID="Assertion-uuidf704f951-0104-f735-d1bb-a0e298abb70b"
IssueInstant="2005-07-08T15:16:35Z" Issuer="https://www.rbtelco.com/rbstocks" MajorVersion="1"
MinorVersion="1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
                <saml:Conditions NotBefore="2005-07-08T15:15:35Z"
NotOnOrAfter="2005-07-08T15:26:35Z">
                    <saml:AudienceRestrictionCondition>
<saml:Audience>urn:itfim-wssm:wsgwsoaphttp1:soaphttpengine:WSGW_BUS:StockQuoteService:wsgw_serv
er1_SOAPHTTPChannel1_InboundPort</saml:Audience>
                    </saml:AudienceRestrictionCondition>
                </saml:Conditions>
                <saml:AuthenticationStatement AuthenticationInstant="2005-07-08T15:16:35Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
                    <saml:Subject>
                        <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">emp1@bigcorp.com</saml:NameIden
tifier>
                    </saml:Subject>
                </saml:AuthenticationStatement>
            </wsse:Security>
    </env:Header>

```



```

<CipherValue>XoQ0E7+cwnSXVJ1epFDb0IZLCzh9Hg4Ghr92sLn9wLziH0zcw7aNew2iBc1jza1P1u1bZVB1/Tmlw4LwnJ
ZK0o1CwGIBfwTAGJ5VmQUUri+105RSwmxrXZTJ+vGTJ319p3rizDz/BPkPz1WuwdBME1Z0kkGvFBUrMnxi49IyvD8=</Cip
herValue>
  </CipherData>
  <ReferenceList>
    <DataReference URI="#wssecurity_encryption_id_1866730207718926501"/>
    <DataReference URI="#wssecurity_encryption_id_766821276426737109"/>
  </ReferenceList>
</EncryptedKey>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
      <ec:InclusiveNamespaces PrefixList="env wsse ds ns1 "
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
    </ds:CanonicalizationMethod>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="#wssecurity_signature_id_3469149583751836238">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <ec:InclusiveNamespaces PrefixList="env ns1 xsi soapenc xsd p680 wsu
soapenv " xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
        </ds:Transform>
      </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <ds:DigestValue>/EJdxNyoawuufENjOPfTna8nyEE=</ds:DigestValue>
  </ds:Reference>
  <ds:Reference URI="">
    <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
        <dsf2:XPath Filter="intersect"
xmlns:dsf2="http://www.w3.org/2002/06/xmldsig-filter2"/*[namespace-uri()='&apos;http://schemas.
xmlsoap.org/soap/envelope/&apos; and
local-name()='&apos;Envelope&apos;]/*[namespace-uri()='&apos;http://schemas.xmlsoap.org/soap/enve
lope/&apos; and
local-name()='&apos;Header&apos;]/*[namespace-uri()='&apos;http://docs.oasis-open.org/wss/2004/01
/oasis-200401-wss-wssecurity-secext-1.0.xsd&apos; and
local-name()='&apos;Security&apos;]/*[namespace-uri()='&apos;urn:oasis:names:tc:SAML:1.0:assertio
n&apos; and local-name()='&apos;Assertion&apos;]"/>
      </ds:Transform>
      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
        <ec:InclusiveNamespaces PrefixList="saml wsse env ds xsi ns1 soapenc
xc14n xsd p680 wsu soapenv " xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <ds:DigestValue>0stczd3sVfs7DVufmsHUawGe8lk=</ds:DigestValue>
  </ds:Reference>
</ds:SignedInfo>

```

```

<ds:SignatureValue>BYbK7XuUjIfES+6x7lZT2JE/nZF0jQKczRbe9CI/nk1xmy+mIjXhYumvODCg7l04P2ikFQ/ged1N
VCjc1V5FsBdxIS37KZ2xzwFCevAbrBEDxw4H9yggw+hxnN43WB2ikn/IJqetkGbYLkM8/des/Wp65kGz5KCORIKPCU8jGHU
=</ds:SignatureValue>
  <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:Reference URI="#x509bst_4298581995360551010"
ValueTypes="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"
/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
  <EncryptedData Id="wssecurity_encryption_id_1866730207718926501"
Type="http://www.w3.org/2001/04/xmlenc#Element" xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
    <CipherData>
<CipherValue>Hwj3Di4hIFlmgJ40NsKuahZehY9hGR7VMLf5ZEUMxzffoVCxGzJdBmFD0QR7b1aubPA1dvTo2aqMHiz+r
nKEOhyQJ/ajG+A3bpBIM0mSdLXQgpVKKqgoF1RX1Sft1NDBYuE339N/SFXotJs2m6Dr9kPPUPYThN3dj0hwi3o7PuihwV9
jHyISzbpR8BVau7bBBtjYI9vMa1QR1wCs1DngiUs137whXDuIx288HbUzMvk27jooM4THCW7Wt0e5i021zLIUyp/VEiCff
wrSzdr5R1841FUx17v/qWIXGxyJa62pw+E9k4bL7iMZ6I5NXAsR8xVFajvRW21hJdtY157Z7njGZtb+E/CjwQyXwFj16X7
2EPeAQgjFw3ikBrB80rtqkBX7cUaiLkWyovhxzAgvUf/GfklFY1C2ekTw/DTLVpevmq4YBw1R8Z4BonHhI41V3jvUK0L1z6
BOADbEB6mJValp8VoufswxM9EeYSb6PZ1WztZePwcyoT9oa701RbiZAUyJ6Lk130pbmzCvEgiKzVYp27yBnrIdwSIFGzRV
todt9XieC4GaJhn/QDixWLTcFKQBS9gfAn6oQcTnv1YQoq11wrSmGpTH1zq4NoRAAPDx0T1dj4xICvg+vPEU27DftzFN8
B3Fe+IGqzIjFubs9bflmZz1e1RXArctiDk9K6mhyTBWxtgY5p1Dc6tX6VPdi6vZia6Eb8jxn18/+NNHicPBCNU1uH0298+b
oY1AATEhXHb34N24djEPiA/s9YmRmUxLRGbp1bMV3pr38HR5PwN16XVYnrGXB1tCLzKAJsqh8AhNnbPetyfSmnF03V1v2ca
91eoLNZK0ed31ixOowcNTW9sPyyA4Y3HuJYZQubm2eWfXEkG5I/w2MkoI8zpDt40zSmho426aQwR1sBbL1CK0hTfHETCrv0
ZSUsn59XNL/JB9sBR+pCAEH6jvg1GXjDiWeKQ1gKMGVZ0/30g1NmXiXB+n/p7WsYaTiWrR/rxs1QD9uk06mt7EQnvzYSrp
iv12/B8N1t/hcG9w8pgyH8StkNeqqpAsjax9iaDHct5L3u7fZnQ1JoAgH1D0tu3Dm0wQA6ScMYIj2XjyUNHR1x2009Rm6A0
8nNwodtaD8/XtF+8D0dkJb6UEQ78PSAfz0EAYFUCVZiwMC6huFswXcFn7LwUuyWSXwinfp88e8LS/nlV81boc+kpgdzF+vy
txD6k0vz2cCeZE6nZ090dv4C4mrhtNpBjoi40/y16T2Ep7uDZzy/Yjm5+ouZ0GhE4KXGjYoUoxqKfz5a08MD0hVkkJmdX7
FZQcR9KSL4wMcy9rKVX10emQzc6Y0unftozD/PnGJM6YDz4GykeJhZFPk5cr4Zta7BHweHzTC4/RyRdZbHctn+mWch8ayK
vRH5yS9ao7se++hKSR2a3LvQWw+SnbjkuV/erftw1GTdZ77E7zpb17pHERb6apWsjMIhnUFMIKaeidKfwJE1oiQXjA1E
1gBpYyfJHhGTEKICJ2UbdHivA5hudXIUvSpEvZ+Z33eW2QnGvDct47Q0pN41uFcufrUuffRaL6P0zEtoFyBdGB6QyH3Zi6
6KZ4G2m7eQ4BSWJmray31a6+tQnKmmHMKQ6nnWrUD/1pN1ygsdjoA01qbBzdIHwyHV+QS50h+XPJQzojaFjHhe8NAudenx
wgVLLVLFchWQZwPEUIz2XNFRSEvytXm/9jqU1cMg5xxAEQFO/nrUm6o0q9LmjKEfAAMEbGT90tYmW8SHL1UnZmZebR35Mw/
SCU8Y1IENS/qMkvTbdJ1tE9Ab/pJS4w5i3rzcLbSef2Lthmu58dkfFoE+HrFKqmN1nw5B8wECf++wGw/OBI6gIv0W2GCUJ/
nKZGwS28K5E01d2i7CFTMJxcSnJMEoNMARrRg2kPCglCVyJ2H822IXvF4MuezPYRAP5031hx0HPiFDQki8oF0doxVti05U
b7JhEps2hCmXm9UVx/vncl84mxPg1NSL14GaA/cdXwPTKkFnaNnMpGdzntIYmlC36KfZkR7v3sajfCns/fkooi9tygR5E
74RNwMCFZRUK7b5UP2vkn0ZrKg9r1hH8JSrQFDX30q50Ry1mtLPY1164ojNjNuzpRLtreYwKegmdHHjeDvgdChBRa0SsafZ
r3hSaIq1iXdwMBXP0GdD1WozNqs4yPs/SyG8613wgm0CUG5N4doEs8wyoLnioti096vcJDaY+y1qLe3VvWvPi1AidRjT8
K7MuV++c28A3vtrk1DDMT4NXV8PTEP1VawRewsWWVJow384EQEcbtCyc4wWL8/HVsV1A0IcChyIZu6VHRTpPJeQvXu7X
sI7j4ClXF3NZGGuHgSujoaELjrLL4EPRH8r/pzg7g7a/0UrpaVwY1cUufdozKhxQNOw0aVFNpRMA415B1UXHyAKMAU7xbo
3QeKoHw1gh5HBU/Od0kxZwpRGndUNJXRKI+a+cIZDeYdJpJxiBjgfv3bz3+Fi8KkR8BC0xNT3sLYCzIqaGu1ws9Xmn99pGc
kN50365R1b3GXddG1TQzpqxCQ8WVRg1ezk8RciPAVz13CW9044YqLgYRifnJADHLXwh11BDJz7XQSL31dZCNI7nAT8K3TRc
uYkdxJEzZR+3v/ep+3K/HMKG9juwEYn/AK/vLjfyiH3Yiga+M2ArP1jSzwn70/A3n0cVpLx4AsL4qJRys/WgAgzQFHk23D6
B1ucP6BUi3VRf5CCczvtBnK7AehTzRX+BY/163B1R9fSXF1RKLURKnCkGqQTXYZONZ4Ezmrrkmt3VYYzDH1QmI71oXuIZ4o
x1DmLcQo/AK62bks1GZM161vrrQigNpZMwmxE20wcw/k1DLHTisqmbOzdYvi/usL3RA1wsF7/ToFhDqLEPeEVQWkxQsp1f9
6KKQyfhQ331y03o5cF+eQ9bIYQOgN8Ph32YRNX7es9J5RNjivn2HTUo+bTma4SH8L7ewrHTwGHqZB50YBZ1ovoYeNX6K/3

```



```

HoeZF+8Kt520kRXn0IenhRKma4w6GskVaPFuLg7VFCd2wYzQX+mrKYKoJcTIm/ILNFD5Zb3xXzi tHkcIPFYx3uVatMz2njb
qAF9j8niCtfc9URFQh09gJu/3Nd3dWwiSdd727m2h43FtKaXTHrpEJ6jGUZV1GLzw9NjW3bQH+muUvER+JjgfiwoDJreRIy
IJdagDPcs6wsxGjW0bhnaCC3ZT/+qQ06+5GJo2LM7smJWsbSsVsZLEFTw3Er16pf51w2MErFsOMM+QCk1uIHdycj61e3i4X
TY1P4fRdm6Yh4qIVFLMrDyAk1vhiyUUZ2yX+F1nDxDuUbyI7i052+ZUW83ekXbAxxAsP/f0t7UWx5sH0cW5VoNnVoxe9Gy
mGTT1HPRNe+S4811xp+d38JK2h8n0i8f9b2AA4Y5JFdv2vzHTMxCJ0scVNHJCvEwa3XvfAjM2rnR86q3Sqc1VgF0y19eQ5A
KkY+wMFPfFNIaFg2hoxuIWGC1SlOkSrpZD+A+HZ4hxnWmj8cSfmVcuNLRI8UgJudedkDhBE1dcfBqrQCmKHwngaJqqS3mM
Q3akfHcpfeQBtIuKutr7oU2jSUJ049tdQp1AtHg+sB2hJ0zad4Wvb/6LUvaCMIVJY0GgoOc+ucIOUBo/oAq1+fbn8SrKQJ
d/+7/Zm8wi8D6X1d619Y1tr8Wc1/hTv0JL9B/ETUQ/TiMMtFN1xPH7K11pda/QBXptpPswm1TsyUuqbVwxtzyhLuk+d3kid
ArqyqRwSwhexbkXu3PtpQ+Fyq5dp7sLKQAakqZEwD5BpS+kFN5c6VT1WFh4MRRwboryKpCtpDQSSuSNnHN40nUzBCkt10Sa
u0GLyUc1aJKDHGsp9po2I+z4nMkmURiOmUWGFbgVWeZzuFuBNx2UbuBGGjxkdp3JJRBVViR2FPVwT9UzJZwoowhB8vfpk
O/oI+XnwM310t4nG0Q6W6UjzrACon0JzdVgODIi2iFjs+Xvi98UDSTdy4Aychg1aeE9edIfCzxVuxXySa6YchE5DhQCaFVI
KXWwXcec3Q6/xMnyxyqKxlyAgsSwyLgdy59Az0xbCri rFH0odBh8o3C3mNNfxUi0m8Bz2FMcXuCgggoITJhvDa38d4eMPm
zbFP4mi63YXEMQXeki8FYE1Qt1mmb1JDYHH02FXF51KFB3rqGkjB9xqmbJLX4BPpD5UT0ZsNU08WAUH10a5JnNBS5YbhYH
4HYYS7fWytcnLYSV4IZS+it5Gvqu3Fxr0+1SXjSMXkP8ZdY4xidcXg1QF+/Hz1qk911gQ44gzIjW+DzWYagYUCu1s80RhxP
QyV3XdGFsMHHX4Ur0gh059ZcZyJM/hVkc0qOpUrsE0F3Fz1IfedVBTS11QsTZbmvwFh/dC1zjo2qp03aEDi9yEx+FHjyP19
BmxMQ1Nzv88XD0+kMzb2aEsfic3MCTIcw8mTxRzJmLu1Dd7LM19im4M12i3UqqDxHqhly6zCqA6Q5PKrNhpYmCAs/WQL7t
np1euZjypQGhvgJXxc/vNJ3buJwlyvL/I1K5Cwp1BfmYUMB/ChtzHVSgiTuh1CYrs/8MDYpdfQGpkUtHo8A==</CipherVa
lue>

```

```

</CipherData>
</EncryptedData>
</wsse:Security>
</env:Header>
<soapenv:Body wsu:Id="wssecurity_signature_id_3469149583751836238"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <EncryptedData Id="wssecurity_encryption_id_766821276426737109"
Type="http://www.w3.org/2001/04/xmlenc#Content" xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
    <CipherData>
<CipherValue>SkW+sGLzRg4dQrwpW5LGfvac/U/PEN8RDszijLumgBM88wt2svCkbls+1P2k07KQCvJAUo0Dq0zqJaK6TR
zdX7nPYCKcuyrKEtJDKrV16SJQMuzUPdjw3urhR1ynC+XP</CipherValue>
    </CipherData>
  </EncryptedData>
</soapenv:Body>
</env:Envelope>

```

10.4.4 Web services security management Token Consumer with SAML Assertion login module

Now we turn our attention to RBStocks and look at how it employs Web services security management to validate the SAML assertion that came across with the Web service request and exchange it for an unsigned assertion that can be used to do a JAAS login to WebSphere. Example 10-7 on page 316 shows a trust

service request to exchange the signed SAML assertion for an unsigned assertion. Note that the trust service also performs blacklist checking, authorization, and identity mapping of the request (to a user called *delayed* or *realtime*).

Example 10-7 SOAP message to trust service providing a signed SAML assertion and requesting an unsigned assertion

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <wst:RequestSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
      <wst:Issuer xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wsa:Address>https://www.rbtelco.com/rbstocks</wsa:Address>
      </wst:Issuer>
      <wsp:AppliesTo xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference>
          <wsa:Address>urn:itfim-wssm:WebProject:services:StockQuoteService</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:Base>
        <saml:Assertion AssertionID="Assertion-uuidf704f951-0104-f735-d1bb-a0e298abb70b"
IssueInstant="2005-07-08T15:16:35Z" Issuer="https://www.rbtelco.com/rbstocks" MajorVersion="1"
MinorVersion="1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
          <saml:Conditions NotBefore="2005-07-08T15:15:35Z"
NotOnOrAfter="2005-07-08T15:26:35Z">
            <saml:AudienceRestrictionCondition>
              <saml:Audience>urn:itfim-wssm:wsgwsoaphttp1:soaphttpengine:WSGW_BUS:StockQuoteService:wsgw_serv
er1_SOAPHTTPChannel1_InboundPort</saml:Audience>
            </saml:AudienceRestrictionCondition>
          </saml:Conditions>
          <saml:AuthenticationStatement AuthenticationInstant="2005-07-08T15:16:35Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
            <saml:Subject>
              <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">emp1@bigcorp.com</saml:NameIden
tifier>
            </saml:Subject>
          </saml:AuthenticationStatement>
        </wst:Base>
      </wst:RequestSecurityToken>
    </soapenv:Body>
  </soapenv:Envelope>

```

```

        <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">empl@bigcorp.com</saml:NameIdentifier>
        </saml:Subject>
        <saml:Attribute AttributeName="user_home"
AttributeNamespace="http://rbtelco.com/user_home">
        <saml:AttributeValue>bigcorp</saml:AttributeValue>
        </saml:Attribute>
    </saml:AttributeStatement>
    <ds:Signature Id="uudf704f958-0104-feba-c2ac-a0e298abb70b"
xmlns:ds="http://www.w3.org/2000/09/xmlsig#">
    <ds:SignedInfo>
    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmlsig#rsa-sha1" />
    <ds:Reference URI="#Assertion-uudf704f951-0104-f735-d1bb-a0e298abb70b">
    <ds:Transforms>
    <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-signature" />
    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
    <xc14n:InclusiveNamespaces PrefixList="saml ds"
xmlns:xc14n="http://www.w3.org/2001/10/xml-exc-c14n#" />
    </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1" />
    <ds:DigestValue>gN6wzAgjvq1TdIZGdKsf0IKC/po=</ds:DigestValue>
    </ds:Reference>
    </ds:SignedInfo>

    <ds:SignatureValue>u0Q3qU6DMiXRnv/9eecVieIz1rfiwgHm03kR4DWDt/nWuHXjLgmb/hnq2driSHpy8AfObLw9kHrxy0wqPp0Yh/UBHFhf47ZeKY5Wnkc0vCdwAh3RxrXBu/ssy9xveqbxGcgqplzDWmufxkNxSvBFvQifQBy1wJQvmhAh0GD8neg
=</ds:SignatureValue>
    <ds:KeyInfo>
    <ds:X509Data>

    <ds:X509Certificate>MIICqjCCAh0gAwIBAgIBCjANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQQDEXNmaW0ucmVkaW9vYm9vY29tMQswCQ...1tPspg8t1VoGEq79uUWj7s+/B5GDjgh92NU6WIOGji0caI31nwIckM38RcBqHPgnHbAiQ0uoxfoss
=</ds:X509Certificate>
    </ds:X509Data>
    </ds:KeyInfo>
    </ds:Signature>
    </saml:Assertion>
</wst:Base>

<wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Validate</wst:RequestType>
    </wst:RequestSecurityToken>
</soapenv:Body>

```

</soapenv:Envelope>

Example 10-8 shows the response from the trust service at RBStocks, which returns an unsigned SAML assertion mapped to either *delayed* or *realtime*. This assertion is used to perform a JAAS login at RBStocks.

Example 10-8 SOAP response from the RBTelco providing an unsigned SAML assertion from which a JAAS login can be performed

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsa:Action
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">http://schemas.xmlsoap.org/ws/2005
/02/trust/RSTR/Validate</wsa:Action>
  </soapenv:Header>
  <soapenv:Body>
    <wst:RequestSecurityTokenResponse wsu:Id="uuidf704fe85-0104-e413-f81c-bc0b9265e970"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <wst:Status>
        <wst:Code>http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid</wst:Code>
      </wst:Status>
      <wst:RequestedSecurityToken>
        <saml:Assertion AssertionID="Assertion-uuidf704fe86-0104-e0d9-428d-bc0b9265e970"
IssueInstant="2005-07-08T15:16:36Z" Issuer="https://www.rbstocks.com/internal" MajorVersion="1"
MinorVersion="1" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
          <saml:Conditions NotBefore="2005-07-08T15:15:36Z"
NotOnOrAfter="2005-07-08T15:21:36Z">
            <saml:AudienceRestrictionCondition>
              <saml:Audience>urn:itfim-wssm:WebProject:services:StockQuoteService</saml:Audience>
            </saml:AudienceRestrictionCondition>
          </saml:Conditions>
          <saml:AuthenticationStatement AuthenticationInstant="2005-07-08T15:16:36Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
            <saml:Subject>
              <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">realtime</saml:NameIdentifier>
            </saml:Subject>
          </saml:AuthenticationStatement>
        </saml:Assertion>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <wss:SecurityTokenReference>
```

```
<wss:KeyIdentifier ValueType="saml:Assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">A
sassertion-uuidf704fe86-0104-e0d9-428d-bc0b9265e970</wss:KeyIdentifier>
  </wss:SecurityTokenReference>
</wst:RequestedAttachedReference>
</wst:RequestSecurityTokenResponse>
</soapenv:Body>
</soapenv:Envelope>
```

10.5 Configuration data

10.5.1 Overall architecture and prerequisites

The Interaction description shown in Figure 10-1, “Use case 4 logical architecture” on page 293 made use of a single WebSEAL instance and three WebSphere Application Server instances. The successful configuration of the scenario depends upon Tivoli Federated Identity Manager Web services security management and all of its prerequisite software having first been installed and configured on the WebSphere Application Servers according to the instructions in the Tivoli Federated Identity Manager Web services security management Guide.

10.5.2 RBTelco configuration

This section presents the configuration of all Web services components at RBTelco, including the Web services client application, the Web services gateway, and the Tivoli Federated Identity Manager trust service.

Use of the WebSphere Stock Quote sample

In order to make the scenario as straightforward as possible for the reader, we base it on the WebSphere Web service Stock Quote sample, which is available from the Samples Gallery within Version 6 of the Rational Software Development Platform. The essential modifications to this sample were:

- ▶ No registration was done to a UDDI registry, as it is not relevant to this use case.
- ▶ Web services client and server extensions and binding configurations were created to fulfill the token and WS-Security signing and encryption requirements of the scenario. On the client hosted at RBTelco this consisted of inserting the Access Manager Binary Security token. On the server hosted at RBStocks this consisted of requiring a SAML assertion token, and requiring the token and message body to be signed and encrypted with standard

WebSphere WS-Security. A key point to recognize here is that while Tivoli Federated Identity Manager is the generator of the SAML assertion at RBTelco and the consumer of the SAML assertion at RBStocks, it is WebSphere WS-Security signing that is responsible for the binding of the SAML assertion to the Web service request.

- ▶ JKS key stores were created and Web service binding configurations were updated so that the custom key stores would be used instead of the default keys that ship with WebSphere. The same signing key was used by the trust service to sign the SAML assertion and by WS-Security to achieve overall message integrity. For details on the key stores that were used in this use case see Appendix C, “Keys and certificates” on page 425.
- ▶ The endpoint of the Stock Quote client JSP was set to the WebSphere Web Services Gateway at RBTelco instead of pointing directly to RBStocks.

Web service client configuration

The first thing that had to be done was to modify the Web Services Client Security Extensions and Client Bindings to configure a Request Generator security token. The role of this Request Generator is to create an Access Manager binary security token for the client making the Web service request to the Stock Quote Web service. Full XML configuration information for the client extension and binding is available in Appendix D, “WS-Security deployment descriptors” on page 437.

Figure 10-8 on page 321 shows the addition of a security token named `AccessManagerToken` to the Web Service Client Security Extensions.

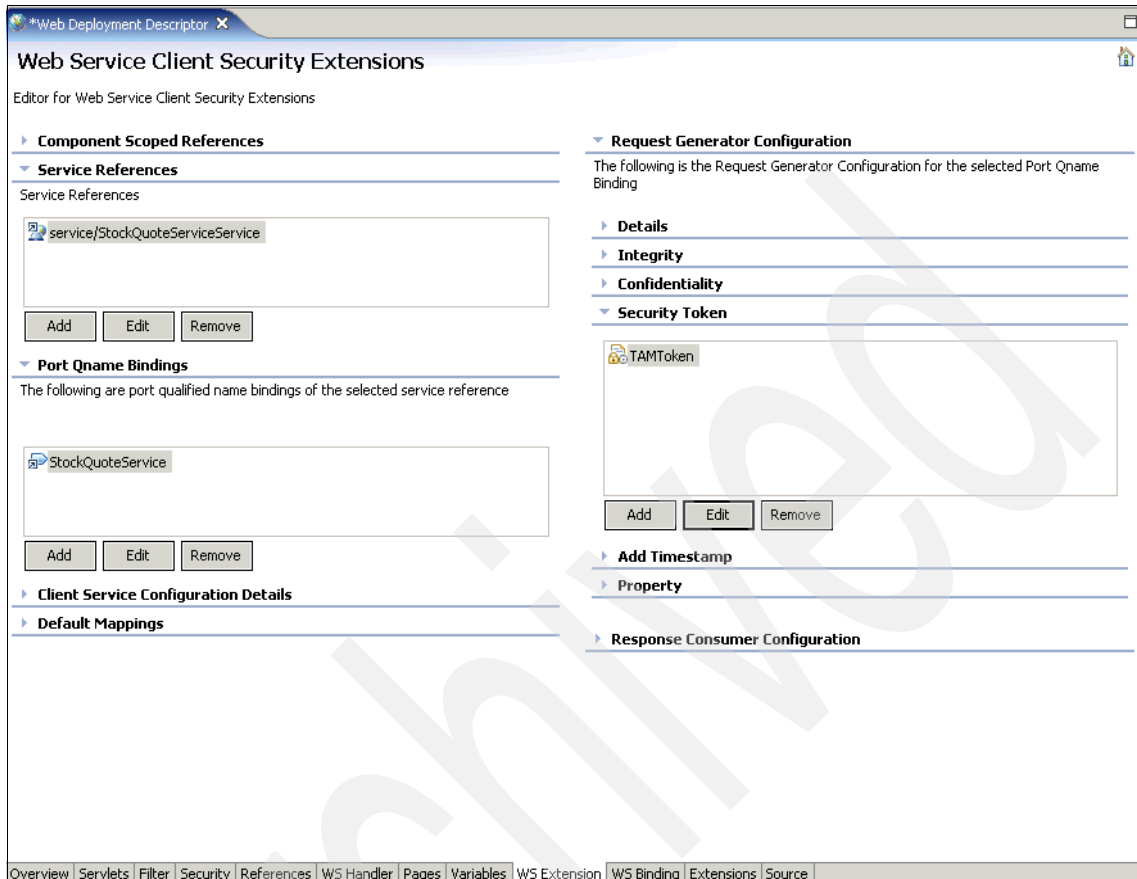


Figure 10-8 Adding Access ManagerToken to the Web Services Client Extensions

Figure 10-9 on page 322 shows the Security Token Dialog box for Access ManagerToken, which displays its URI and Local name.

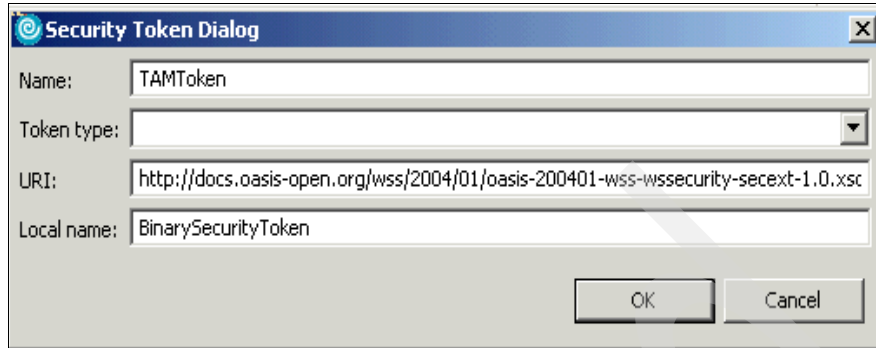


Figure 10-9 Security Token Dialog for Access ManagerToken

The next thing that had to be configured was the Web Service Client Bindings for the token. Figure 10-10 on page 323 shows the addition of a Token Generator named TAMTokenGenerator to the Security Request Generator Binding Configuration.

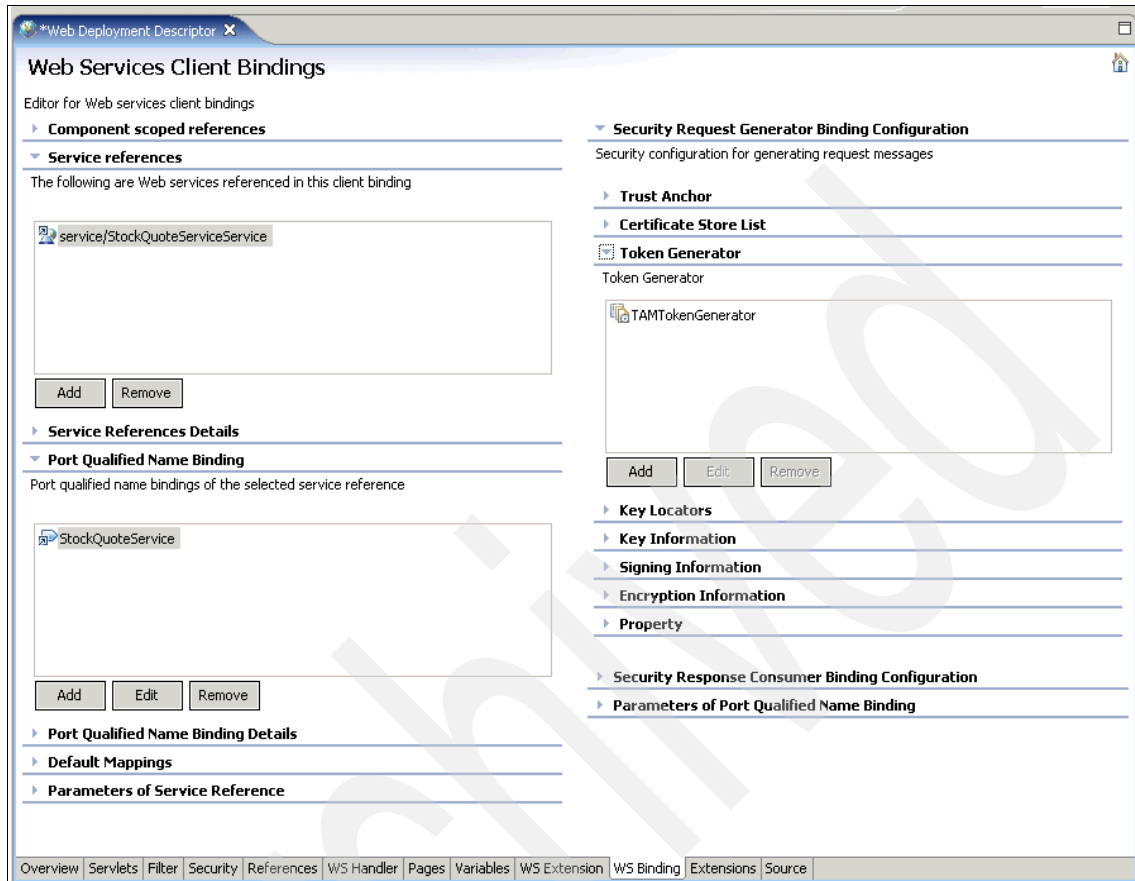


Figure 10-10 TAMTokenGenerator added to the Security Request Generator Binding Configuration

The Token Generator Dialog box for TAMTokenGenerator is shown in Figure 10-11 on page 324. It describes a WSSMTOKENGenerator with a callback handler of `com.tivoli.am.fim.wssm.callbackhandlers.TAMTAICallbackHandler`. It also specifies a single property of `trust.service.call`, which in this case is set to `false`, as no call out to the trust service is required at the client for this scenario.

The callback handler is responsible for locating the Access Manager credentials of the authenticated user and generating an Access Manager binary security token. The TAMTAICallbackHandler does this by inspecting the current JAAS security subject associated with the user for an Access Manager credential contained within a PDPrincipal. The PDPrincipal is inserted into the JAAS subject during the WebSphere TAI++ login via WebSEAL.

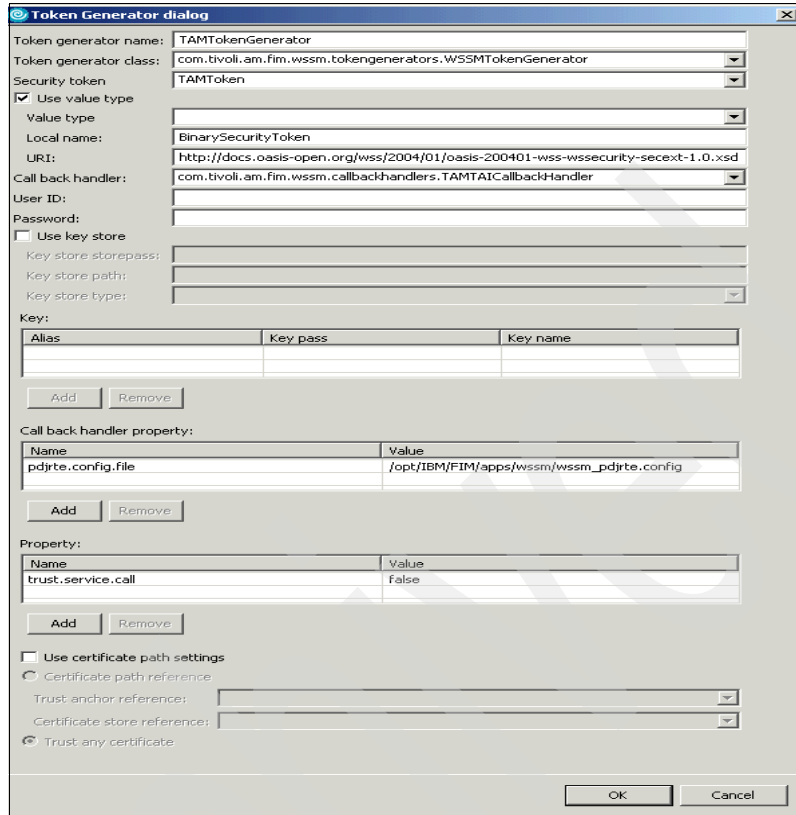


Figure 10-11 Token Generator Dialog for TAMTokenGenerator

Additional changes required to the Web service client

In order to utilize the TAMTAICallbackHandler provided by Web services security management, it was necessary to provide the client application with Java security permissions, as shown in Figure 10-12.

```
grant codeBase "file:${application}" {
    permission javax.security.auth.AuthPermission "wssecurity.getCallerSubject";
    permission javax.security.auth.PrivateCredentialPermission "*" * "*" * "read";
};
```

Figure 10-12 was.policy update granting use of getCallerSubject

WebSphere global security settings on the gateway

Before Web Service Security Extensions and Bindings can be configured for the application at the WebSphere Web Services Gateway, global security settings on the gateway had to be updated. It is this update that allows the WS-Security

Bindings to reference the Access Manager binary security token login module, which will log the user into the WebSphere Web Services Gateway.

Figure 10-13 shows the addition of `itfim.wssm.tamcredential` to the System login configuration. In the figure you will also see an entry for `itfim.wssm.saml`, which should generally be added to the system login configuration during Web services security management deployment, but that is not used at the gateway in our scenario. It will, however, be used at the RBStocks WebSphere Application Server when an SAML assertion is used to authenticate to WebSphere.

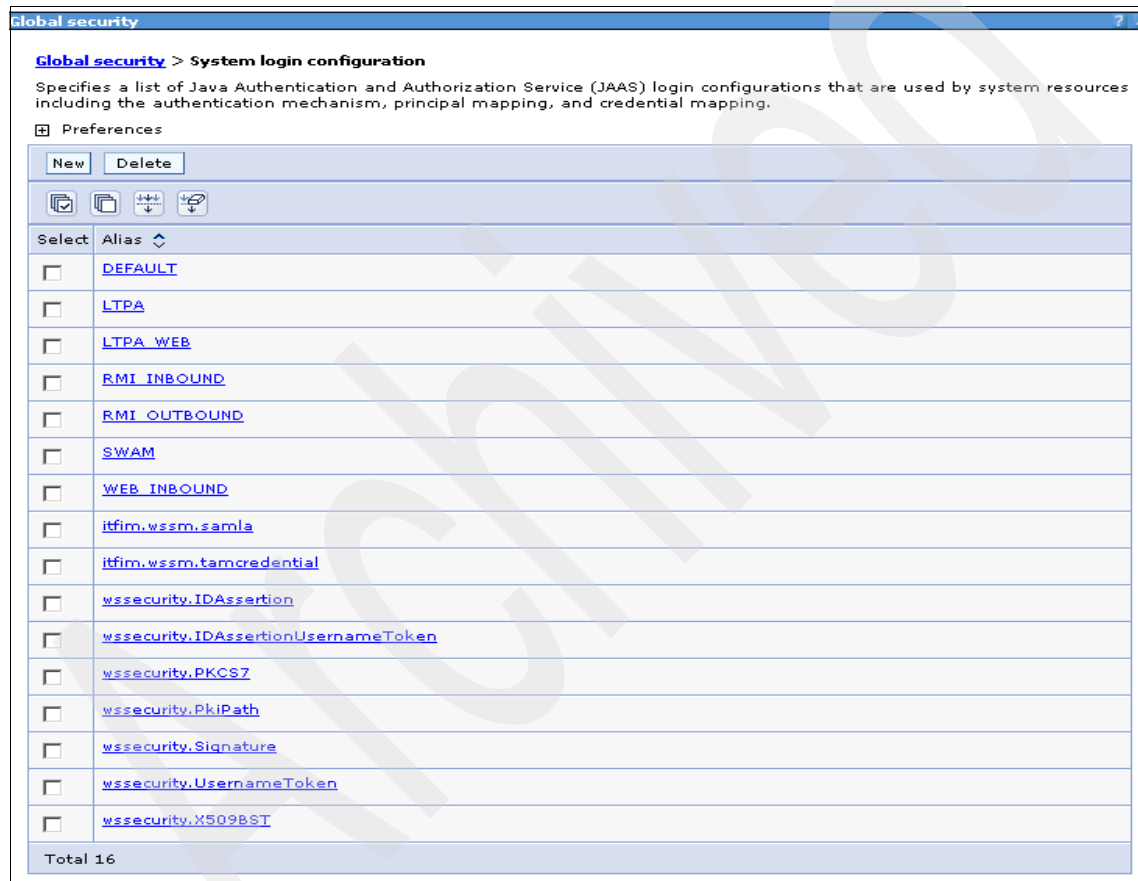


Figure 10-13 System login configuration showing that `itfim.wssm.tamcredential` is available

The `itfim.wssm.tamcredential` entry is expanded in Figure 10-14 on page 326.

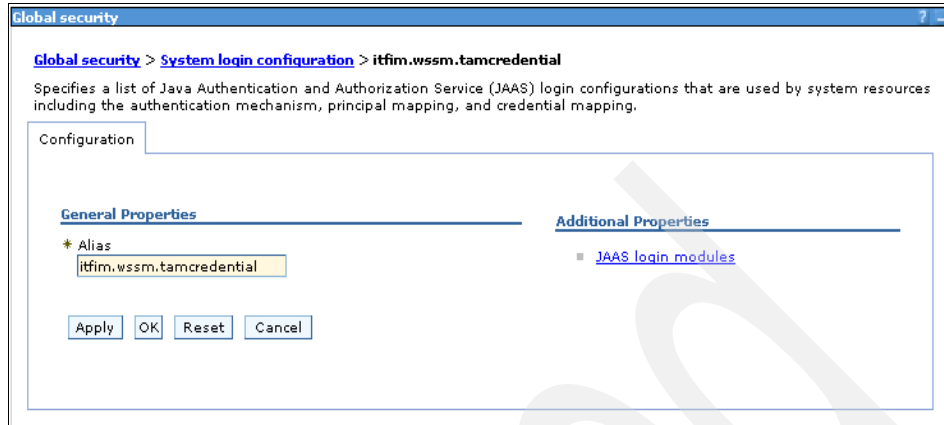


Figure 10-14 itfim.wssm.tamcredential

Figure 10-15 shows the JAAS login module configuration for itfim.wssm.tamcredential.

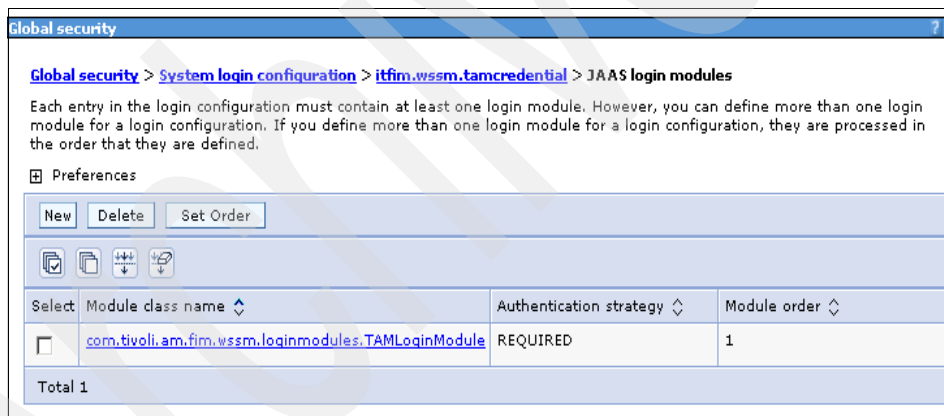


Figure 10-15 JAAS login modules for itfim.wssm.tamcredential

Figure 10-16 on page 327 shows the general properties for the configured JAAS login module.

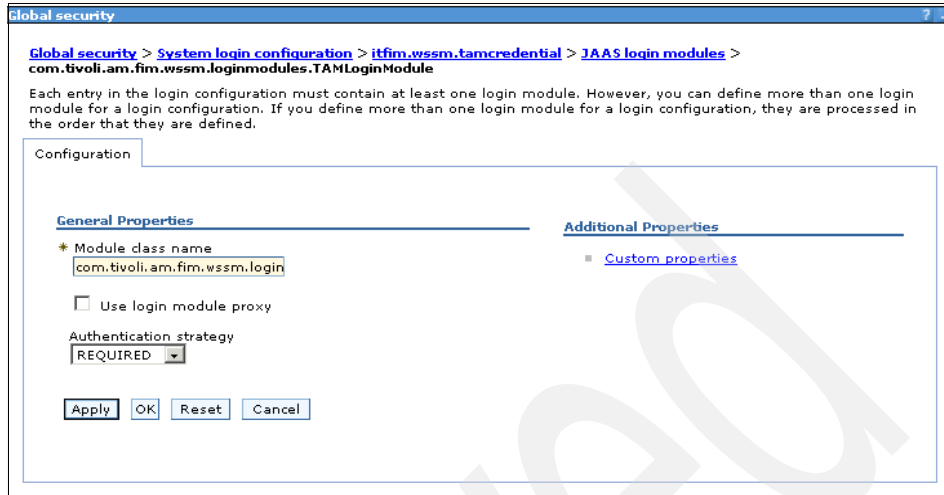


Figure 10-16 General properties for the itfim.wssm.tamcredential login module

Inbound Web services gateway configuration

Figure 10-17 shows the list of WS-Security configurations on the gateway. These are the WebSphere Web Services Gateway equivalent to the WS-Security Extensions that would be configured on a WebSphere Application Server. The entry of interest here is StockQuoteServiceInboundFinal, which is configured as Service Type *Inbound*. The full XML configuration file information for this extension is available in “RBTelco WSGW server extension configuration” on page 440.

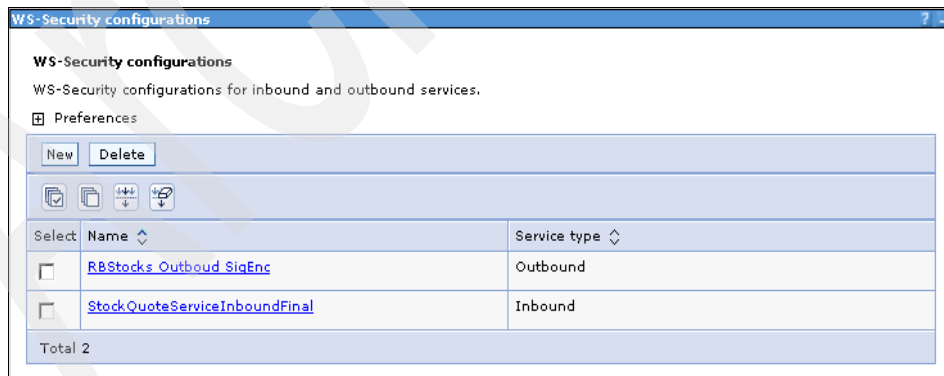


Figure 10-17 WS-Security configurations for use case 4

Figure 10-18 on page 328 is the expansion of the StockQuoteServiceInboundFinal entry.

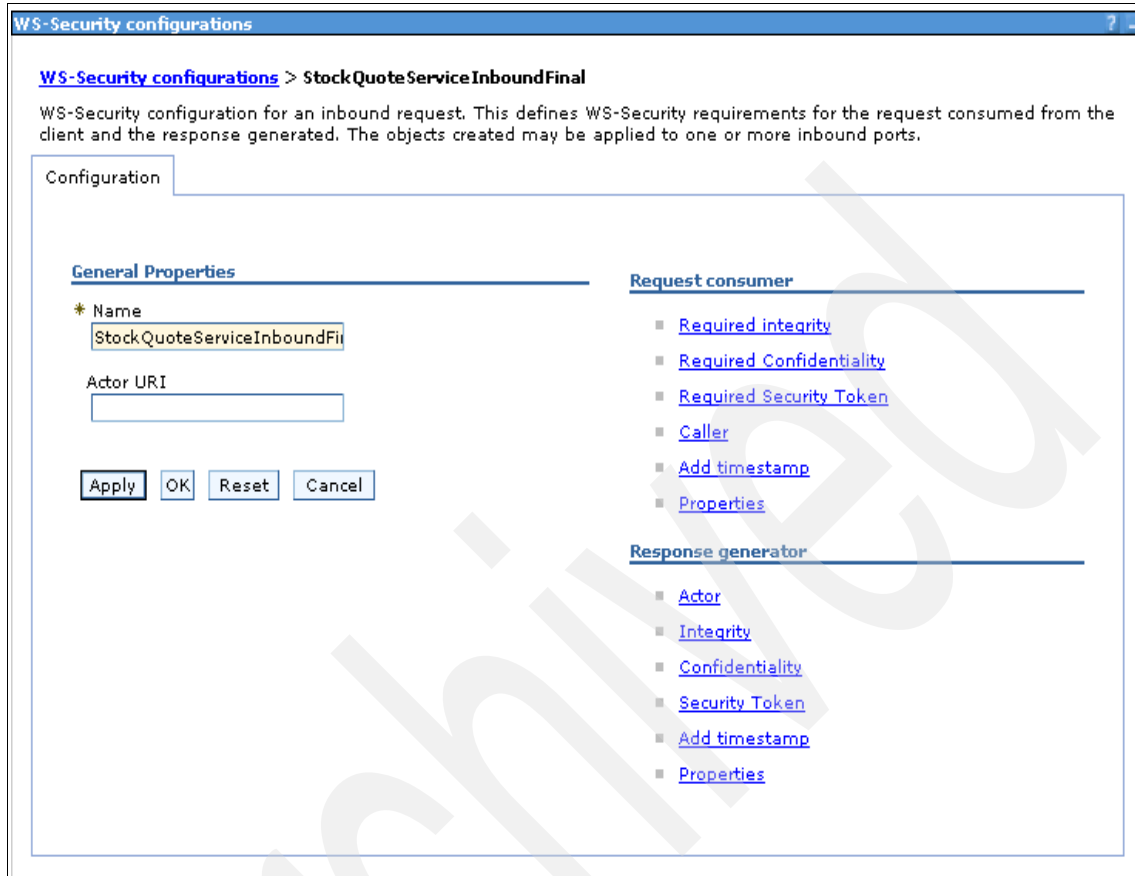


Figure 10-18 WS-Security configuration for the inbound service StockQuoteServiceInboundFinal

Configured for the *request consumer* was a *required security token* and *caller*. The configuration of the required security token for TAMCredential is shown in Figure 10-19 on page 329.





WS-Security configurations

[WS-Security configurations](#) > [StockQuoteServiceInboundFinal](#) > **Required Security Token**

Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller. Standard and custom security tokens may be defined by URI and local name. A UsernameToken is defined with a URI of <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and the local name of UsernameToken. An X509 token is defined with a URI <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and a local name of X509v3. A Kerberos token is defined with a URI of <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and a local name of Kerberosv5ST. A SAML token is defined with the URI of <urn:oasis:names:tc:SAML:1.0:assertion> and a local name of assertion. An XrML token is defined with a URI of <urn:mpeg:mpeg21:2003:01-REL-R-NS> and a local name of license. A SecurityContextToken is defined with a URI of <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and a local name of SecurityContextToken. If the usage assertion is Required then the required security token must be in the incoming message. If Optional is specified then both messages with or without the required security token are accepted. The default usage assertion is Required.

⊞ Preferences

New Delete

Select	Name ↕	Usage ↕	URI ↕	Local name ↕
<input type="checkbox"/>	TAMCredential	Required		http://ibm.com/2004/01/itfim/ivcred
Total 1				

Figure 10-19 Required security token for TAMCredential

The expansion of the TAMCredential entry from Figure 10-19 is shown in Figure 10-20 on page 330.



Figure 10-20 TAMCredential specified as required security token (the complete Local name, which could not be shown, is <http://ibm.com/2004/01/itfir/ivcred>)





A Caller had to be configured. The expansion of the Caller entry on the WS-Security configuration screen for TAMCredential is shown in Figure 10-21 on page 331.

WS-Security configurations > **StockQuoteServiceInboundFinal** > **Required Security Token** > **TAMCredential** > **Caller**

Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication. Standard and custom security tokens may be defined by URI and local name. A UsernameToken is defined with a URI of <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and the local name of UsernameToken. An X509 token is defined with a URI <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and a local name of X509v3. A Kerberos token is defined with a URI of <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and a local name of Kerberosv5ST. A SAML token is defined with the URI of <urn:oasis:names:tc:SAML:1.0:assertion> and a local name of assertion. An XrML token is defined with a URI of <urn:mpeg:mpeg21:2003:01-REL-R-NS> and a local name of license. A SecurityContextToken is defined with a URI of <http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> and a local name of SecurityContextToken.

⊞ Preferences

New Delete

Select	Name	Part	URI	Local name
<input type="checkbox"/>	TAMCredential			http://ibm.com/2004/01/itfim/ivcred

Total 1

Figure 10-21 Caller configuration for TAMCredential

Further expanding the TAMCredential entry above would show that Properties and a Trust Method could be configured for a Caller. We do not illustrate that here, as these settings are not applicable to the TAMCredential.

Now that we have seen the WS-Security configuration for the inbound processing of the request, we need to turn to the associated WS-Security bindings. *StockQuote Service Request Consumer Final* is the name of the WS-Security binding for the Request Consumer on the inbound side of the gateway. The entry is shown in Figure 10-22 on page 332. The full XML configuration of the binding is available in “RBTelco WSGW server binding configuration” on page 441.



Figure 10-22 WS-Security binding for StockQuoteService Request Consumer Final

Figure 10-23 on page 333 shows that a single WSSMTOKENConsumer named TAMCredential has been configured.

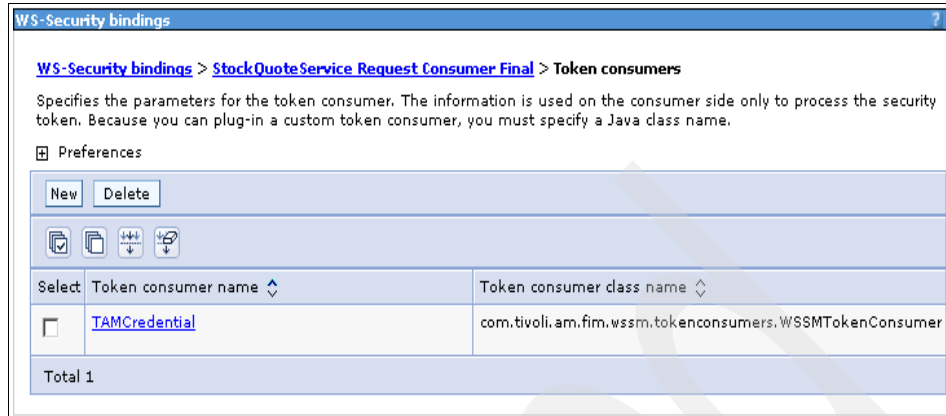


Figure 10-23 TAMCredential Token Consumer

Figure 10-24 on page 334 shows the General and Additional properties page for TAMCredential.

WS-Security bindings > StockQuoteService Request Consumer Final > TAMCredential

Specifies the parameters for the token consumer. The information is used on the consumer side only to process the security token. Because you can plug-in a custom token consumer, you must specify a Java class name.

Configuration

General Properties	Additional Properties
* Token consumer name TAMCredential	■ JAAS configuration
* Token consumer class name com.tivoli.am.fim.wssm.token	■ Properties
Part reference name TAMCredential	
Certificate path <input checked="" type="radio"/> None <input type="radio"/> Trust any <input type="radio"/> Dedicated signing information Trust anchor <input type="text" value="(none)"/> Certificate store <input type="text" value="(none)"/>	
Trusted ID evaluators <input checked="" type="radio"/> None <input type="radio"/> Existing evaluator definition <input type="text" value="(none)"/> <input type="radio"/> Binding evaluator definition * Trusted ID evaluator name <input type="text"/> * Trusted ID evaluator class name com.ibm.wsspi.wsssecurity.id.	
Username token <input type="checkbox"/> Verify nonce <input type="checkbox"/> Verify timestamp	
Value type * Local name http://ibm.com/2004/01/itfir	

Figure 10-24 Properties for the TAMCredential Token Consumer

In Figure 10-25 on page 335 the JAAS configuration for TAMCredential is displayed.

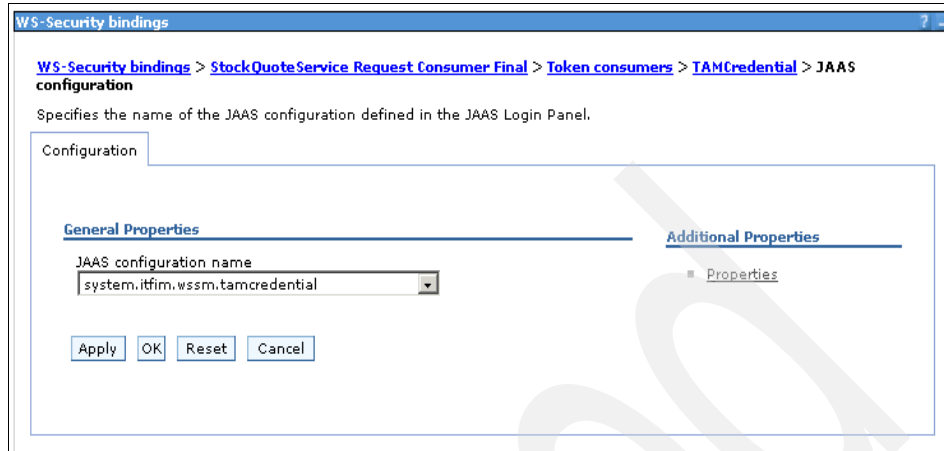


Figure 10-25 JAAS Configuration for TAMCredential Token Consumer

It is here that the relationship between the WS-Security configurations and Bindings and the Global settings that were previously configured becomes clear. system.itfim.wssm.tamcredential had to first be configured in the system login configuration before it could be referenced in the WS-Security Binding.

Figure 10-26 shows the required properties for the JAAS configuration.



Figure 10-26 JAAS Configuration properties for TAMCredential

As shown above, this must point to a valid AZN API configuration file.

Figure 10-27 on page 336 shows the additional properties that were configured.

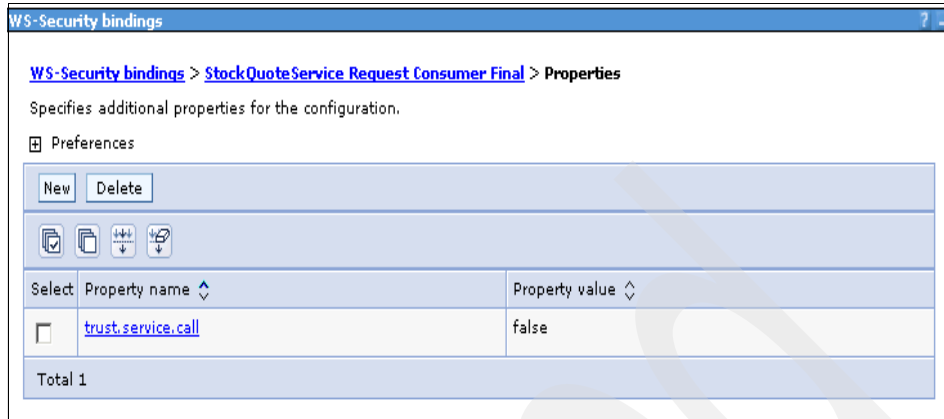


Figure 10-27 Additional properties for the TAMCredential Request Consumer

The only additional property set is `trust.service.call`, and this is set to `false`, as there is no call to the trust service required on the inbound side of the gateway. We are going to log in directly with the token presented in the incoming message.

This completes the description of the configuration on the inbound side of the WebSphere Web Services Gateway. With this configuration a client can be logged into the gateway with the Access Manager binary security token that is sent across in the security token section of the Web service request from the WebSphere Application Server hosting the Stock Quote Web service client.

10.5.3 Outbound Web services gateway configuration

In the section we begin to describe the trust service configuration required to generate an SAML assertion, but first, as in the two previous sections, we need to describe how the WS-Security configurations (Extensions) and Bindings were set.

Figure 10-28 on page 337 shows the relevant WS-Security configuration that has been named *RBStocks Outbound SigEnc* to reflect that this is the outbound request from the gateway to RBStocks and that WS-Security has been used to sign and encrypt the request.

A full XML version of this configuration information can be located in “RBTelco WSGW client extension configuration” on page 442.



Figure 10-28 WS-Security configuration for RBStocks Outbound SigEnc

While the request generator above is configured for both integrity and confidentiality of the Web service request, we only show the configuration of the security token in this document. For information on how integrity and confidentiality are configured for a Web-Service request, please see "WebSphere Application Server Version 6.0 Information Center" on page 472.

Figure 10-29 on page 338 shows the SAML Security token associated with RBStocks Outbound SigEnc. This says that the outbound request from the gateway will contain a SAML Assertion security token.

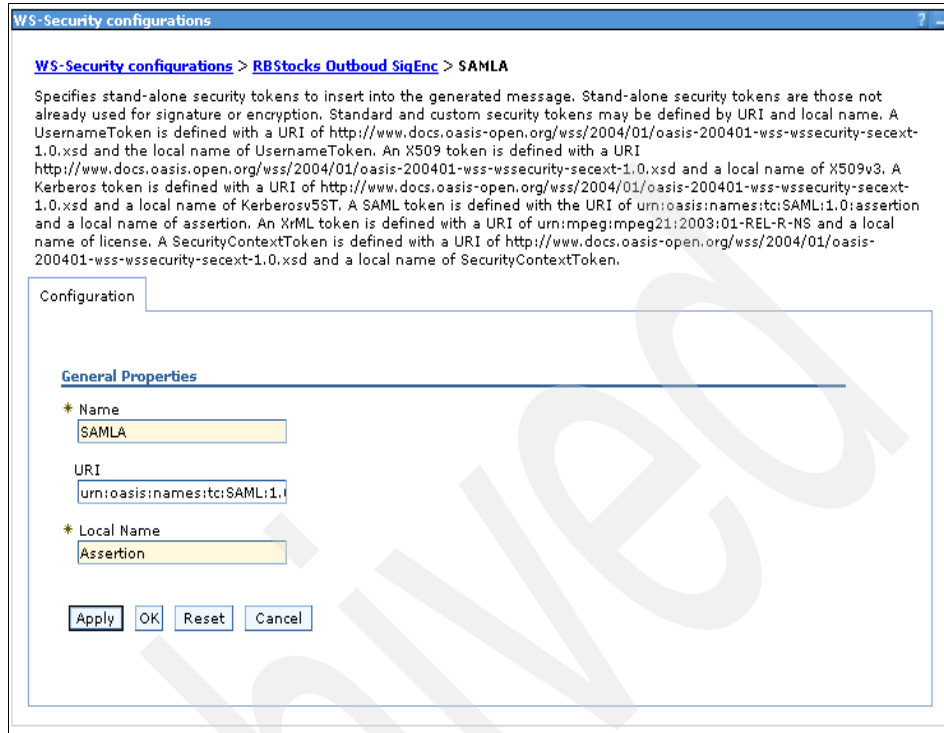


Figure 10-29 SAML as security token type for RBStocks Outbound SigEnc

The complete URI, which is not visible above, is urn:oasis:names:tc:SAML:1.0:assertion.

Figure 10-30 on page 339 shows the related WS-Security bindings. The particular binding of interest is *RBStocks Request Generator SigEnc*.

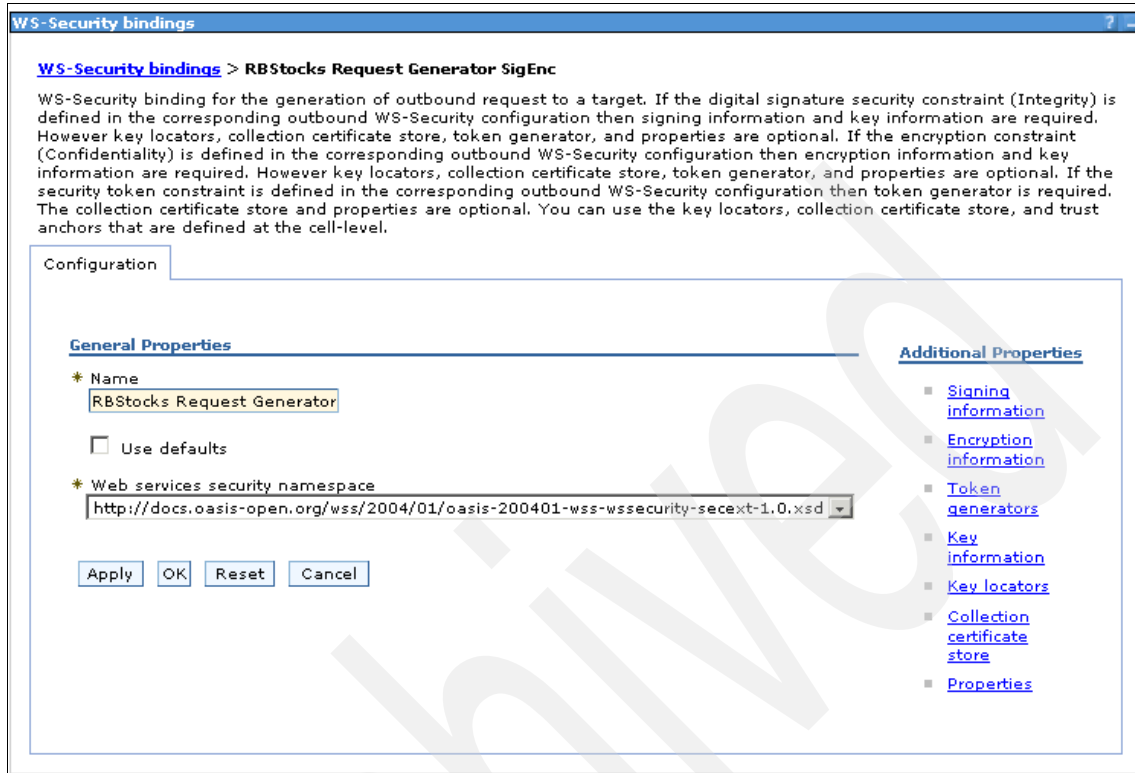


Figure 10-30 Binding configuration for outbound side of the gateway

Figure 10-31 on page 340 shows the two token generators that were associated with RBStocks Request Generator SigEnc.

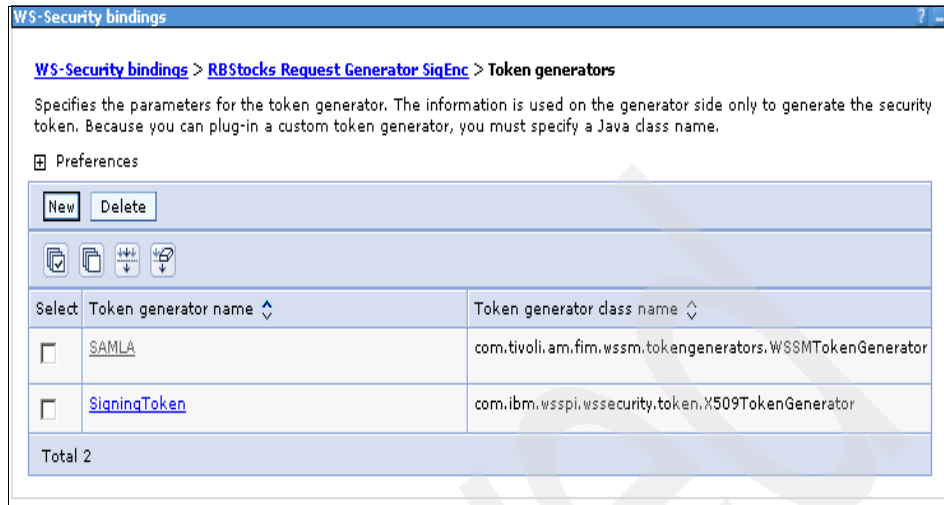


Figure 10-31 Token generators for RBStocks Request Generator SigEnc

The token generator that we need to take a closer look at is SAML, which generates the SAML assertion security token that will be inserted as the security token in the Web service request before it leaves the gateway. The General and Additional properties page for SAML is shown in Figure 10-32 on page 341.

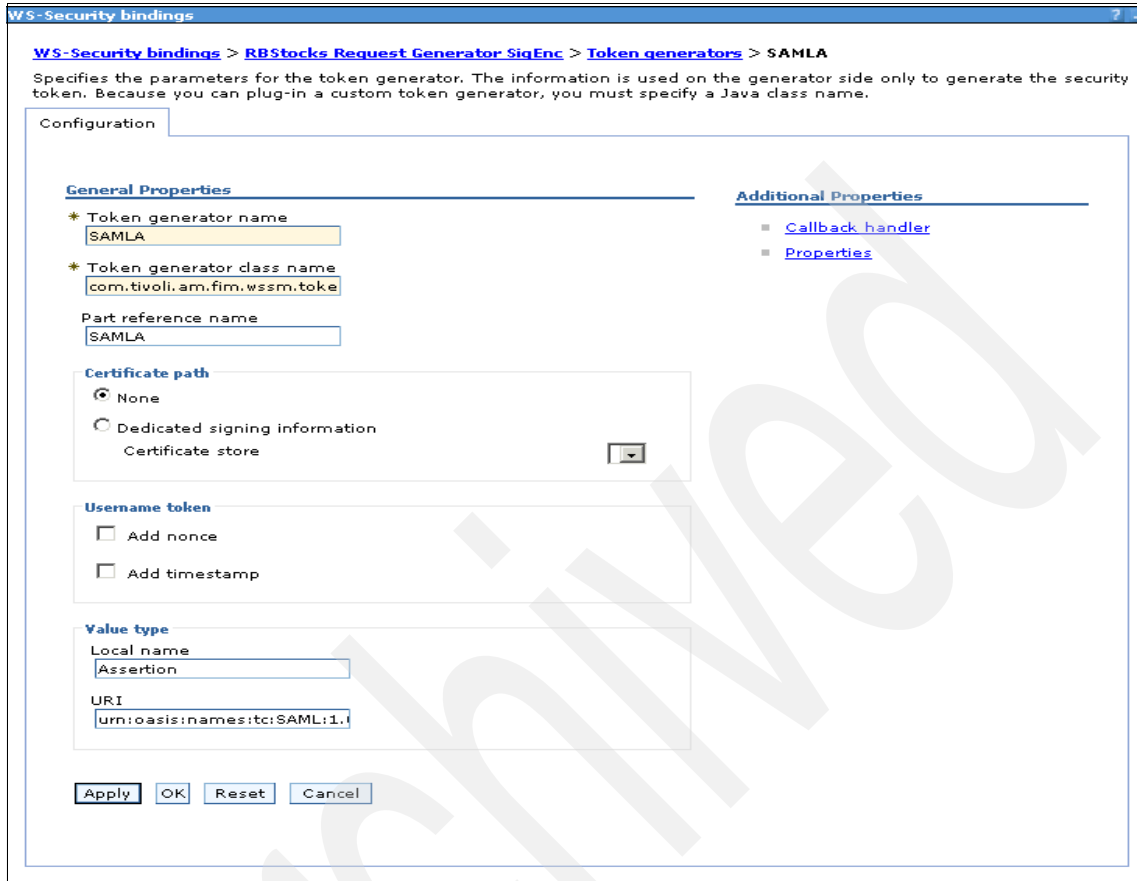


Figure 10-32 General and Additional properties for SAML2

The token generator class name, which is not visible above, is `com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator` and the URI is `urn:oasis:names:tc:SAML:1.0:assertion`.

Figure 10-33 on page 342 shows the properties that were configured for the SAML2 token.



Figure 10-33 Properties for the SAML token

The properties shown in Figure 10-33 specify how Web services security management will interface with the trust service.

- ▶ `trust.service.call` is set to true so that the trust service will be called.
- ▶ `trust.service.url` specifies the location of the trust service.
- ▶ `default.issuer.uri` tells the trust service who the partner is that is making the trust service call. It becomes the Issuer URL in the trust service call and it must match the value that is configured for the Partner Provider ID configured in the Tivoli Federated Identity Manager console. This will be further clarified later.

Figure 10-34 on page 343 shows the callback handler that was configured for the SAML token.

WS-Security bindings

WS-Security bindings ?

[WS-Security bindings](#) > [RBStocks Request Generator SigEnc](#) > [Token generators](#) > [SAML](#) > **Callback handler**

Specifies the parameters for the callback handler that are used for generating the token. Because you can plug-in a custom callback handler, you must specify the implementation class name. WebSphere Application Server provides options for identity assertion, basic authentication, and the key store that are passed to the callback handler implementation.

Configuration

<u>General Properties</u>	<u>Additional Properties</u>
<p>* Callback handler class name <input type="text" value="com.tivoli.am.fim.wssm.callb"/></p> <p>Identity assertion</p> <p><input type="checkbox"/> Use identity assertion</p> <p><input type="checkbox"/> Use RunAs identity</p>	<ul style="list-style-type: none"> ▪ Keys ▪ Properties
<p>Basic authentication</p> <p>User ID <input type="text"/></p> <p>Password <input type="text"/></p>	
<p>Key store</p> <p>Password <input type="text"/></p> <p>Path <input type="text"/></p>	

Figure 10-34 Callback handler for the SAML token

The truncated classname for the callback handler is `com.tivoli.am.fim.wssm.callbackhandlers.WSSMCallbackHandler`.

There are no other properties or keys that are configured on this page. Using the `WSSMCallbackHandler` allows us to retrieve the TAM credential from the JAAS login context that was inserted on the inbound side of the gateway with the `TAMTAILoginModule`. This will then be exchanged for the SAML assertion at the trust service according to the token generators configuration.

Trust service configuration at RBTelco

At this point we have covered the WS-Security configurations and bindings required to use a SAML assertion as the security token on the outbound side of the gateway and now we move on to describe the trust service configuration that supports this. It is here that some familiarity with the Tivoli Federated Identity Manager Web Services Security Management Guide would be particularly beneficial. Below are the two `wsdl2tfim` commands that generated the Access Manager object space and trust service application module chain that were used in this scenario. The syntax and options for these commands can be found in the Web Services Security Management Guide.

`./wsdl2tfim.sh -action config-tam -w StockQuoteWSGW <gateway WSDL file>` was run on the server hosting the trust service to create a Tivoli Access Manager object space of `StockQuoteWSGW`. The generated object space is shown in Example 10-9.

Example 10-9 Access Manager object space created by `wsdl2tfim config-tam`

```
/itifim-wssm
  /wssm-default
    /StockQuoteWSGW
      /StockQuoteService
        /StockQuoteService
          /getQuote
```

RBTelco could use this object space to restrict which of its clients can access the service at the gateway. RBTelco opted not to do this. It is entirely up to RBStocks to determine who is authorized to call the service. When we look at the Access Manager object space generated by the `wsdl2tfim config-tam` command at RBStocks, we see that there is an Access Manager ACL policy associated with it.

`./wsdl2tfim.sh -action config-fim -t SAML11Module <gateway WSDL file>` was then run on the same server to create the application trust chain.

The display of the trust service module chains shown in Figure 10-35 on page 345 shows that the command completed successfully.

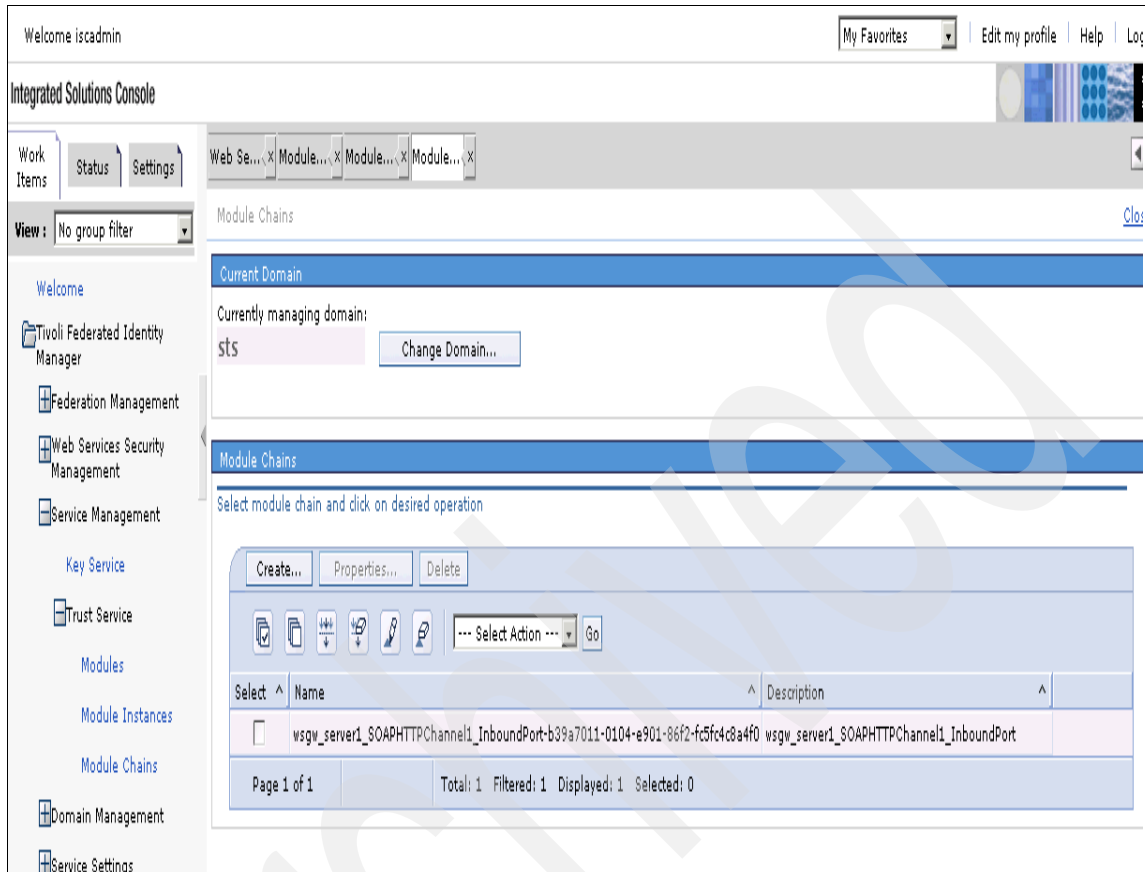


Figure 10-35 Trust service application module chain at RBTelco

The next thing that had to be done was to create a partner module instance. The module instance was named *StockQuote client*, and its type was set as *IVCredModule*, as the trust service will be receiving an Access Manager credential. It is displayed in the Tivoli Federated Identity Manager Console under Service Management → Trust Service → Module Instances, as shown in Figure 10-36 on page 346.

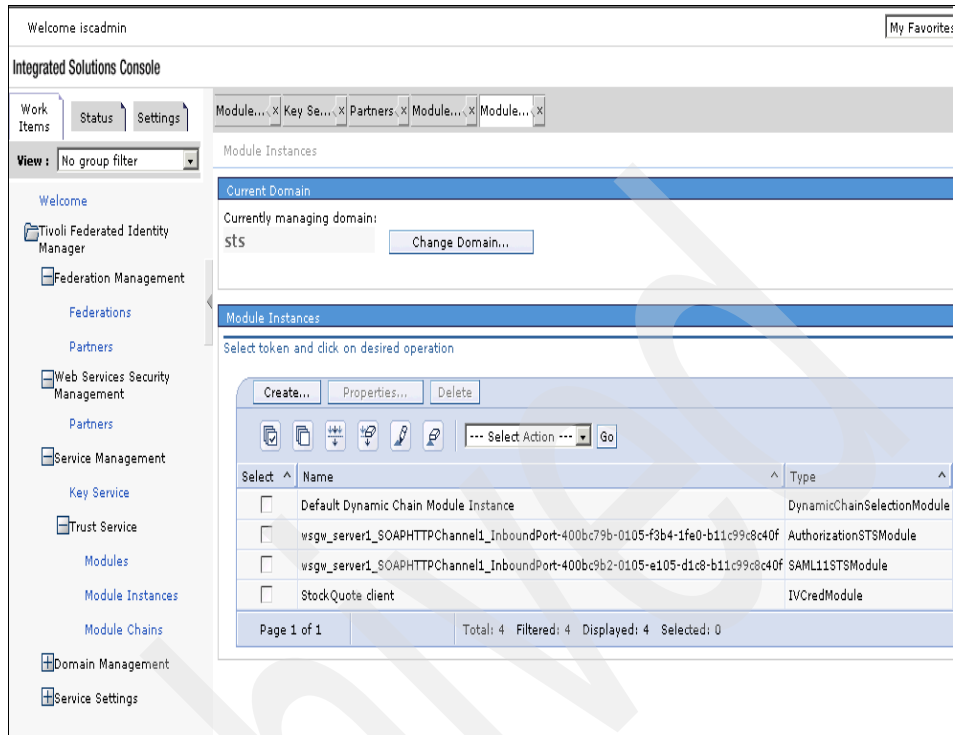


Figure 10-36 Module instances at RBTelco

Looking at the module instance properties for *Stock Quote client* in Figure 10-37, note that it was configured with a valid Tivoli Access Manager configuration file.

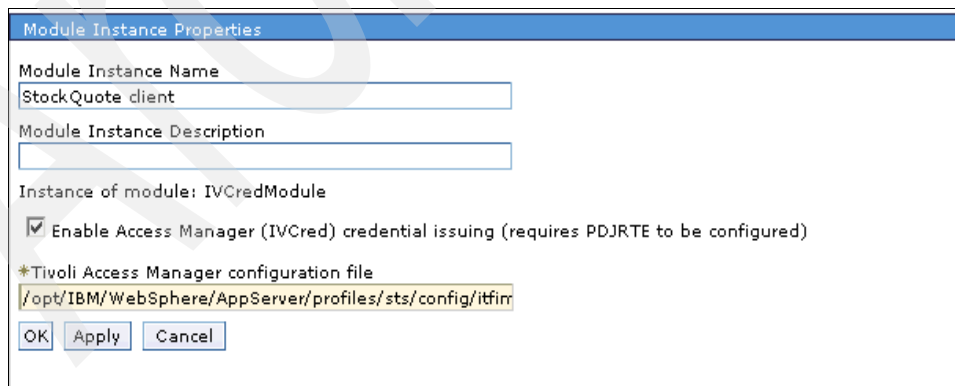


Figure 10-37 Module properties for Stock Quote client

A partner then had to be configured. This can be seen in the Integrated Solutions Console by expanding the partner entry name of RBTelco under Web services security management/partners, as shown in Figure 10-38.

The screenshot shows a dialog box titled "Web Services Security Partner Properties". It contains several sections:

- *Provider ID:** A text field containing "http://www.rbtelco.com/internal".
- *Company Name:** A text field containing "RBTelco".
- Company URL:** An empty text field.
- Contact Person:** A section with four text fields: "First Name", "Last Name", "Email Address", and "Phone Number", all of which are empty.
- Access Manager Credential (IVCred) Module Configuration:** A section containing:
 - An unchecked checkbox labeled "Enable signature validation".
 - A text field labeled "*Select Validation Key" containing an empty value.
 - A button labeled "View/Edit Identity Mapping Rule...".
 - Three buttons at the bottom: "OK", "Apply", and "Cancel".

Figure 10-38 Partner properties for company name RBTelco

Note that the provider ID matches the value configured for the default.issuer.uri shown in Figure 10-33 on page 342. An identity mapping rule was configured when creating the partner. The mapping rule is not shown here, but can be found in "RBTelco mapping for use case 4" on page 418.

The final step in configuring the trust service to support the exchange of an Access Manager credential for an SAML assertion on the outbound side of the gateway was to go to the properties for the module chain shown in Figure 10-35 on page 345, locate the module chain item for *exchange*, and then enable signing of assertions specifying the appropriate signing key. This is shown in Figure 10-39 on page 348.

Module Chain Item Properties

Security Assertion Markup Language (SAML) Module Configuration

Enter the required values to configure the SAML Module

The name of the organization issuing the assertions

*Amount of time the assertion is valid after being issued (seconds)

Enable the Signing of Assertions

Select Key for Signing Assertions

Include the following attribute types (a "*" means include all types)

Figure 10-39 Module chain properties for module chain item of type exchange

Notice the key used for signing is `rbtelco-signing_rbtelco_rbstocks`. An explanation of the keys and strategy for naming them can be found in Appendix C, “Keys and certificates” on page 425.

With the trust service configuration complete, the outbound side of the gateway can now generate a signed SAML assertion and set it as the security token in the Web service request to RBStocks.

10.5.4 RBStocks configuration

This section presents the configuration of all Web services components at RBStocks, including the Web services server application and the Tivoli Federated Identity Manager trust service.

RBStocks application configuration

First, as in the previous section, the WS-Security extensions and bindings are described. The RBStocks application utilized WS-Security signing and encryption over the SAML assertion token and message body to ensure confidentiality of the message, and bind the security token to the message body. This section, however, concentrates on those parts of the configuration related to the SAML assertion token, and the callout to the trust service to validate it and exchange it for a mapped (unsigned) SAML assertion. Full XML reproduction of

the actual extension and binding configurations as used at RBStocks is available in “Web services server RBStocks” on page 448.

Figure 10-40 shows the required security token and caller part in the Request Consumer section of the Web Service Security Extensions.

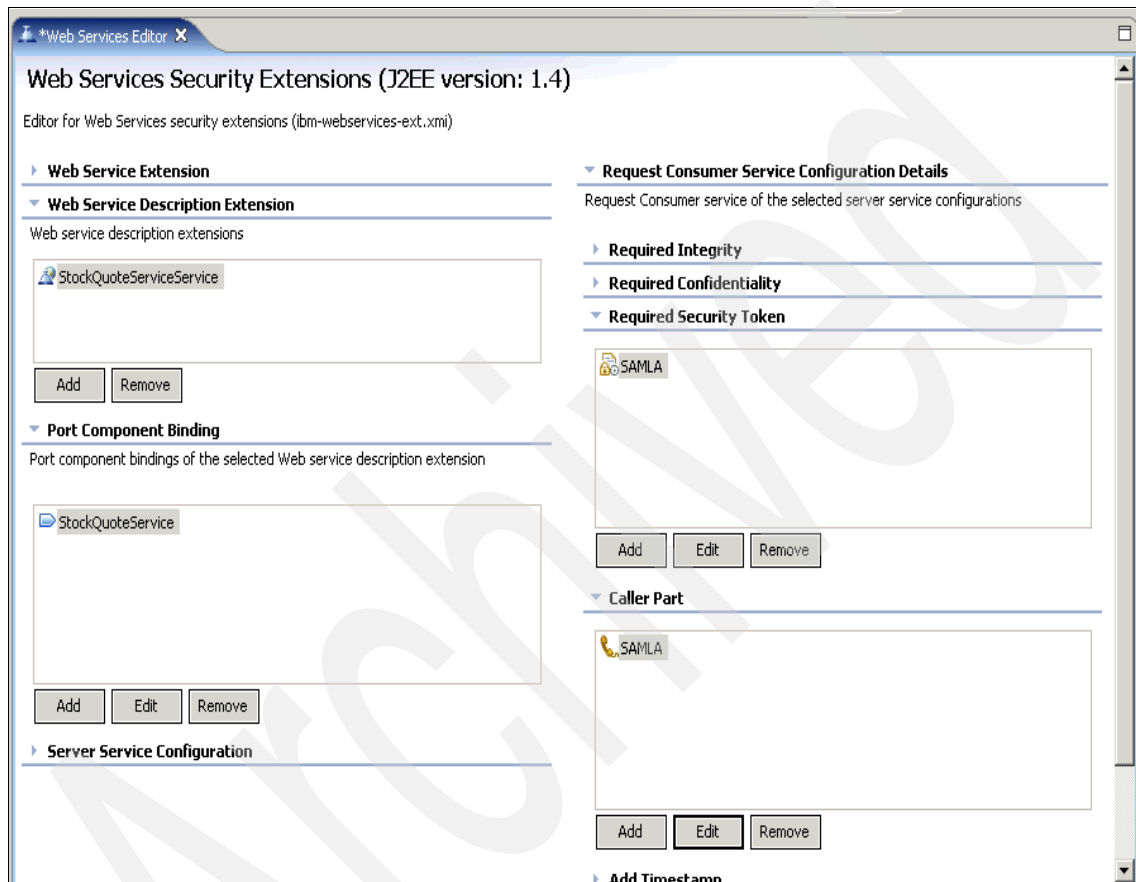


Figure 10-40 Web Services Security Extensions at RBStocks

The SAML security token and Caller part are each expanded below in Figure 10-41 on page 350 and Figure 10-42 on page 351.

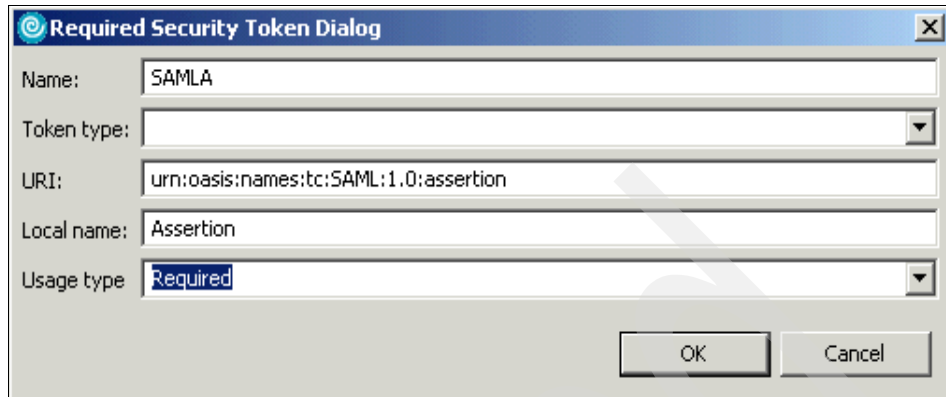


Figure 10-41 Security Token Dialog for SAML2 at RBStocks

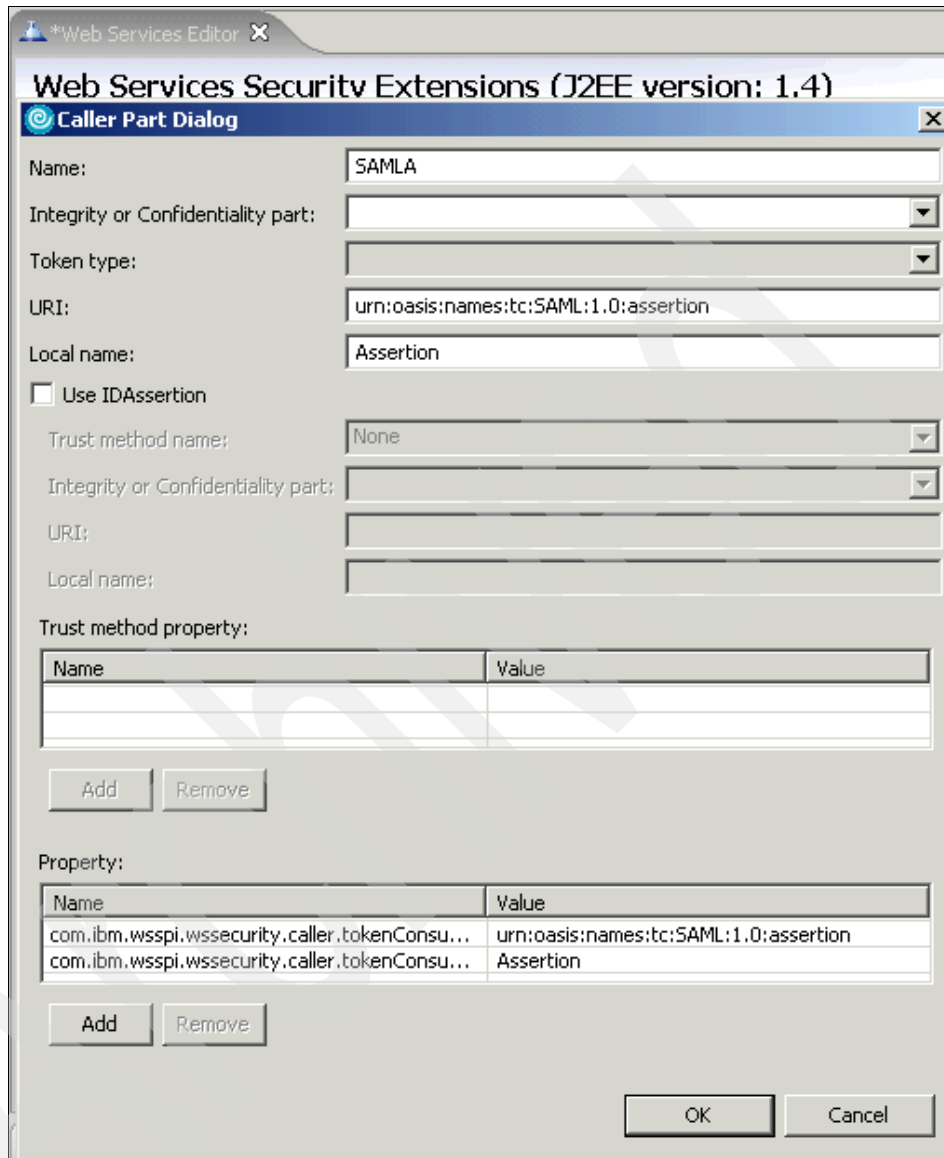


Figure 10-42 Web services extension Caller Part at RBStocks

The property names and values from the figure above are com.ibm.wsspi.wssecurity.caller.tokenConsumerNS with a value of urn:oasis:names:tc:SAML:1.0:assertion and com.ibm.wsspi.wssecurity.caller.tokenConsumerLN with a value of Assertion.

In Figure 10-43 the Web Services binding configurations at RBStocks are displayed.

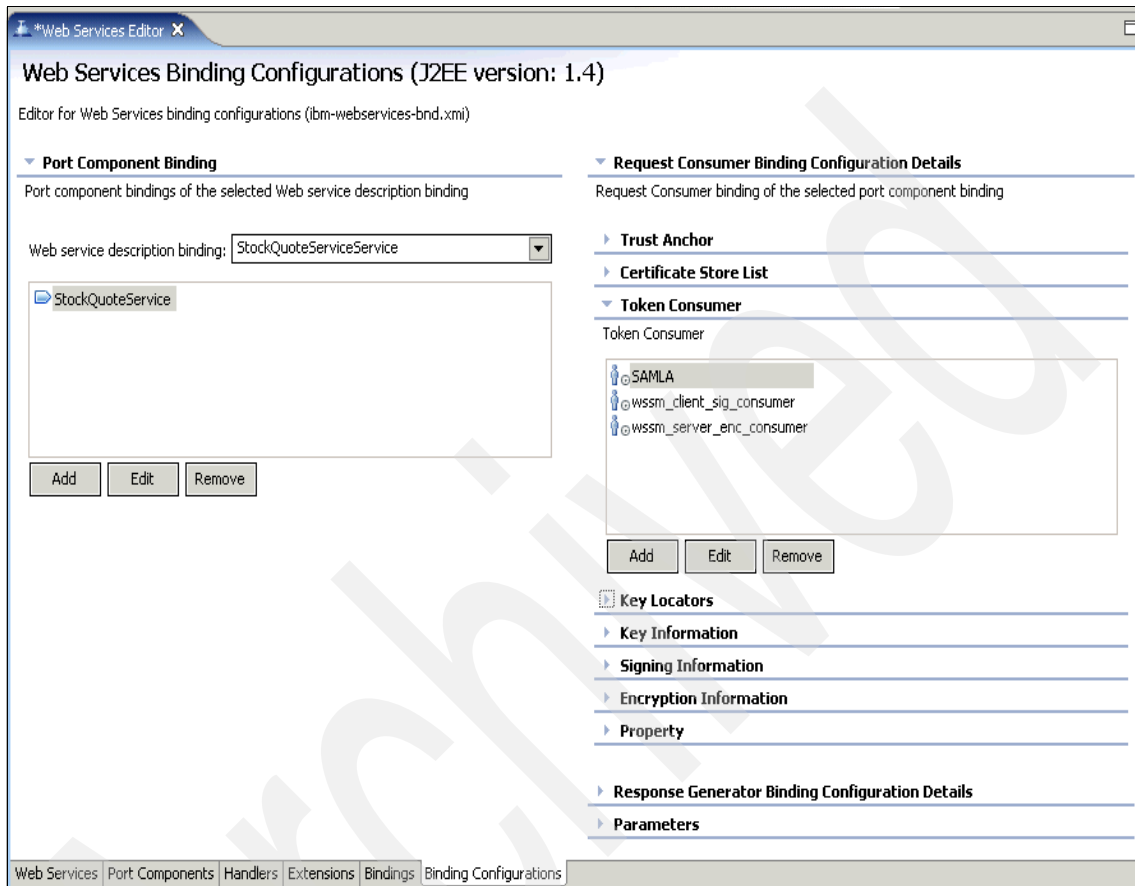


Figure 10-43 Web Services Binding Configurations at RBStocks

There are three Request Consumer Token consumers listed above. Two are for the XML signing and encryption. The security token consumer is SAML, which is expanded in Figure 10-44 on page 353.

Token consumer name:

Token consumer class:

Security token:

Use value type

Value type:

Local name:

URI:

Use jaas.config

jaas.config name:

jaas.config property:

Name	Value

Use trusted ID evaluator

Trusted ID evaluator class:

Trusted ID evaluator property:

Name	Value

Use trusted ID evaluator reference

Trusted ID evaluator reference:

Property:

Name	Value
trust.service.call	true
trust.service.url	http://www.rbstocks.com:9082/TrustServer/Sec...

Use certificate path settings

Certificate path reference

Trust anchor reference:

Certificate store reference:

Trust any certificate

Figure 10-44 SAML security token at RBStocks

In the properties above, trust.service.call is set to true, as a trust service call is required to transform the signed SAML assertion that was received into an

unsigned assertion. trust.service.url is also set to specify the location of the trust service that will be called.

Now that the extensions and bindings have been configured, we need to look at the trust service configuration that supports this. Below are the two `wsdl2tfim` commands that generated the Access Manager object space and trust service application module chain. The syntax and options for these commands can be found in the Web Services Security Management Guide.

`./wsdl2tfim.sh -action config-tam -w StockQuoteWSDL <Stock Quote WSDL file>` was run on the server hosting the trust service at RBStocks to create a Tivoli Access Manager object space of StockQuoteWSDL. The generated object space is shown in Example 10-10.

Example 10-10 StockQuoteWSDL object space at RBStocks

```
/itfim-wssm
  /wssm-default
    /StockQuoteWSDL
      /StockQuoteServiceService
        /StockQuoteService
          /getQuote
```

There are three users defined in Access Manager for this use case. The users *realtime* and *delayed* have been granted access to the service. The user *blacklist* does not have access to the service. When the mapping rules are discussed, we will see that all users accessing the Web service are mapped to one of these three users.

The Access Manager policy used to protect this Web service is shown in Example 10-11.

Example 10-11 Access Manager policy for Web service at RBStocks

```
group create stockusers cn=stockusers,o=rbstocks,c=us
group modify stockusers add realtime
group modify stockusers add delayed

acl create stockusers
acl modify stockusers set user sec_master TcmdbsvaBR1
acl modify stockusers set group stockusers T[WebService]i
acl attach /itfim-wssm/wssm-default/StockQuoteWSDL stockusers
```

To create the application trust chain on RBStocks, we ran:

```
./wsdl2tfim.sh -action config-fim -t SAML11Module <Stock Quote WSDL file>
```


The display of the trust service module chain shown in Figure 10-45 shows that the command completed successfully.

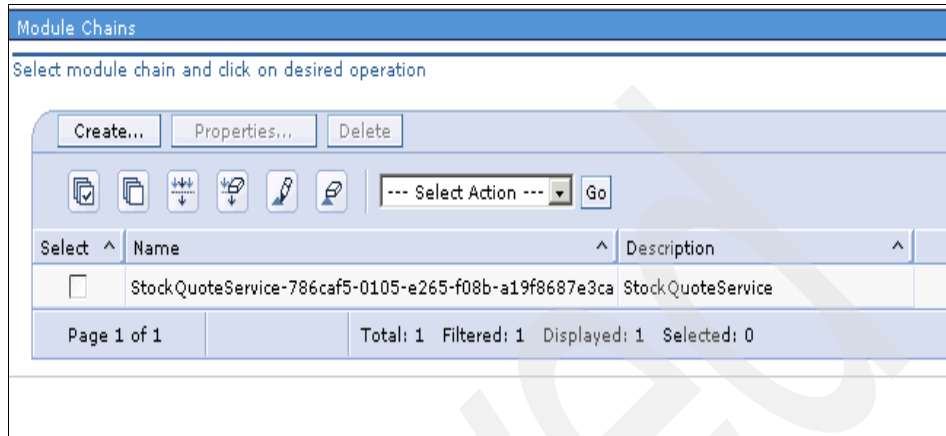


Figure 10-45 Trust service module chain at RBStocks

The next thing that had to be done was to create a partner module instance. The module instance was named *RBTelco SAML assertion*, and its type was set as *SAML11STSMModule*, as the trust service will be receiving a SAML 1.1 assertion. Its properties can be displayed in the Tivoli Federated Identity Manager Console under Service Management → Trust Service → Module Instances, as shown in Figure 10-46 on page 356.

Current Domain
Currently managing domain: sts Change Domain...

Web Services Security Partner Properties

*Provider ID
https://www.rbtelco.com/rbstocks

*Company Name
RBTelco

Company URL

Contact Person

First Name Last Name

Email Address

Phone Number

Security Assertion Markup Language (SAML) Module Configuration

Enable Signature Validation

*Select Validation Key
rbstocks-partners_rbtelco_rbstocks

View/Edit Identity Mapping Rule...

OK Apply Cancel

Figure 10-46 Partner properties at RBStocks

Note that in Figure 10-46 signature validation is enabled and the appropriate public key was selected to perform the validation. For more information on the keys used for our scenarios, please see Appendix C, “Keys and certificates” on page 425.

A mapping rule was also used when creating the partner. Refer to “RBStocks mapping for use case 4” on page 420 to see the XSL mapping at RBStocks. This uses Java code to check a text blacklist as part of the mapping. This is a many-to-few mapping where every user of this service is mapped into one of the three users previously mentioned: realtime, delayed, or blacklisted.

The final step in configuring the trust service to support the exchange of a signed SAML assertion for an unsigned SAML assertion from which a JAAS login can be performed was to go to the properties for the module chain shown in Figure 10-45 on page 355, locate the module chain item for *other*, and enable authorization checks. This causes the trust service to interface with Access Manager to determine a user’s access to invoke the Web service and is shown in Figure 10-47 on page 357.

Module Chain Item Properties

Web Service Authorization Module Configuration

Enter the required values to configure the Web Service Authorization Module

Enable authorization checks

Principal attribute name

Principal attribute type

OK Cancel

Figure 10-47 Module chain item properties for module chain item type of other

10.6 Troubleshooting

This section describes the locations of log files and 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, included where in the configuration to specify endpoints for directing traffic via TCPMON.

10.6.1 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 may be different depending upon your installation directory):

```
-Dcom.tivoli.am.fim.svc.config.location=/opt/IBM/FIM/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, we recommend that you edit this file and change the property traceLevel to the value DEBUG_MAX.

The trace file will appear in:

```
/opt/IBM/FIM/logs/tivoli-common/FBT/wssm/logs/trace.log
```

Again, the path may vary for your installation.

10.6.2 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 utilized by the Web services security management components, you can modify the trace level using the Tivoli Federated Identity Manager Console. Navigate to **Service Settings** → **Logging Settings**, and you will see the logging page. Change the Trace Level to Maximum to enable the most detailed trace. Figure 10-48 shows this screen.

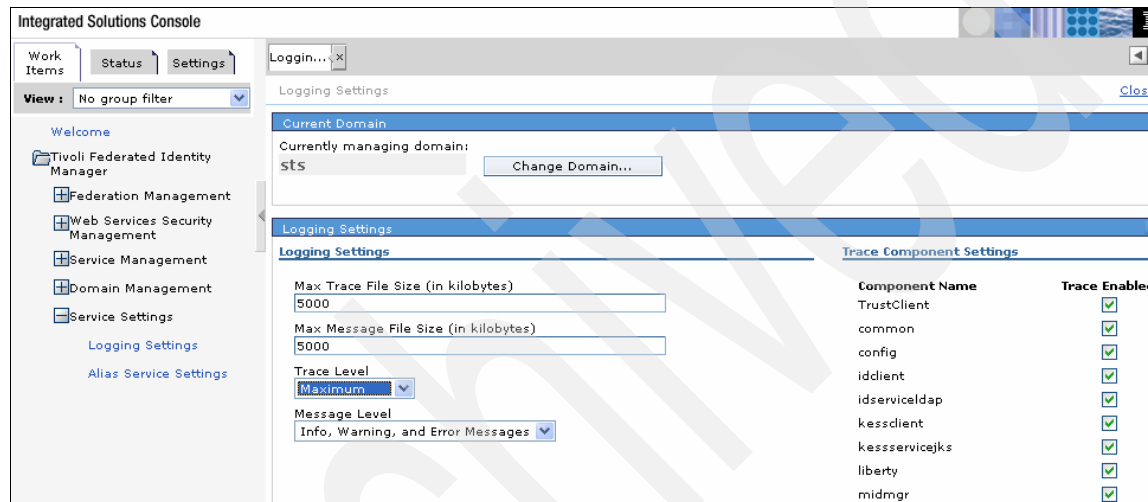


Figure 10-48 Trace settings for Tivoli Federated Identity Manager Runtime

10.6.3 Using the WebSphere logs

The most common log file to inspect in WebSphere is the server's stdout log file called SystemOut.log. It can be found at:

```
/opt/IBM/WebSphere/AppServer/profiles/<profilename>/logs/server1
```

Again, this path may vary depending upon your particular installation. This file will generally provide high-level information about application startup and critical errors. For an extremely detailed trace, set the trace setting in the server (using the WebSphere administration console) to:

```
*=info: com.ibm.ws.sib.webservices.*=all: com.ibm.wsspi.webservices.*=all:  
com.ibm.ws.webservices.*=all: com.ibm.wsspi.wssecurity.*=all:  
com.ibm.ws.wssecurity.*=all: com.ibm.xml.soapsec.*=all:  
com.ibm.ISecurityUtilityImpl.*=all:  
com.ibm.ISecurityLocalObjectTokenBaseImpl.*=all:
```

```
com.ibm.ISecurityLocalObjectBaseL13Impl.*=all: ORBRas=all: SASRas=all:  
com.ibm.ws.security.*=all: com.ibm.wsspi.security.*=all
```

To set this trace string, use the WebSphere administration console and navigate to **Troubleshooting** → **Logs and Trace** → **server1** → **Change log level details**.

Figure 10-49 shows a partial shot of this screen.

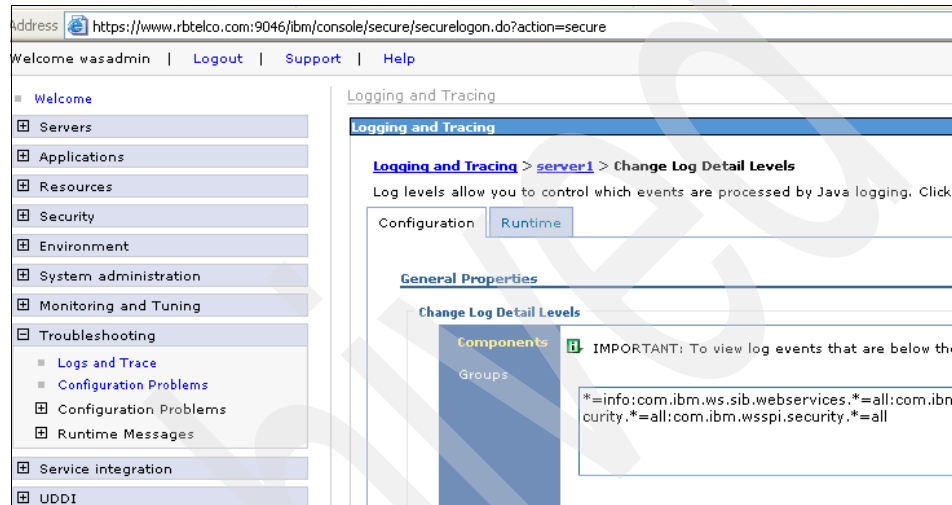


Figure 10-49 Setting detailed WebSphere trace

The resultant trace.log file can be found in the same directory as SystemOut.log.

10.6.4 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 destination. There is a lot of information 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 our scenario.

There are four locations in the Web services scenario where use of TCPMON would be relevant to see where messages are flowing. Figure 10-1 on page 360 presents these flows, and where to configure the endpoint to point to TCPMON.

Table 10-1 Using TCPMON in the Web services scenario

Web services message flow	Configuring an endpoint for TCPMON
<p>The Web services request originating from the JSP Web services client enroute to the gateway.</p>	<p>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.</p>
<p>Call to trust service from outbound side of Web services gateway to exchange Access Manager credential for signed SAML assertion.</p>	<p>Modify the trust.service.url parameter in the Token Generator configuration of the Web services gateway outbound binding configuration.</p>
<p>Call from gateway to RBStocks. This is a signed and encrypted message, so there is not a lot to see.</p>	<p>Modify the service URL in the WSDL file being read by the gateway, and re-import the WSDL into the gateway.</p>
<p>Call from the RBStocks application server to the trust service to exchange the signed SAML assertion for a mapped, unsigned SAML assertion.</p>	<p>Modify the trust.service.url parameter in the Token Consumer configuration of the Stock Quote application.</p>

Appendixes

The following appendixes give a detailed description of various federation configuration subjects that are common to the applications of the federation scenario, introduced in Part 2, “Customer environment” on page 181.

In Appendix A, “Configuring Access Manager WebSEAL and Web plug-in” on page 363, we describe how to configure Tivoli Access Manager WebSEAL and the Web Plug-ins for Access Manger for integration with Tivoli Federated Identity Manager.

Appendix B, “Identity mapping rules” on page 381, shows an approach to authoring the XSL identity mapping rules for Tivoli Federated Identity Manager, and also contains all of the identity mapping rules used in the scenarios in this book.

Appendix C, “Keys and certificates” on page 425, describes the keys and certificates that were generated for the use cases described in this book.

Appendix D, “WS-Security deployment descriptors” on page 437, contains the WS-Security deployment descriptors used at the various WS-Security integration points for Chapter 10, “Use case 4 - Web services security management” on page 291.

Archived



Configuring Access Manager WebSEAL and Web plug-in

In this appendix we describe how to configure Tivoli Access Manager WebSEAL and the Web plug-ins for Access Manger for integration with Tivoli Federated Identity Manager.

Introduction

This appendix describes in detail the steps necessary to configure Tivoli Access Manager WebSEAL or the Web plug-ins to interact with Tivoli Federated Identity Manager. We will assume that a stock Access Manager installation is already in place, and so will not deal with the specifics of installing and configuring Access Manager.

The Tivoli Federated Identity Manager software uses exactly the same on-the-wire integration interfaces with both WebSEAL and the Web plug-ins, the difference being that no junction is involved with Web plug-ins.

Figure A-1 shows a logical deployment architecture with WebSEAL. Tivoli Federated Identity Manager runs as a junctioned application, typically in a separate WebSphere environment (cluster or single server) from other business applications.

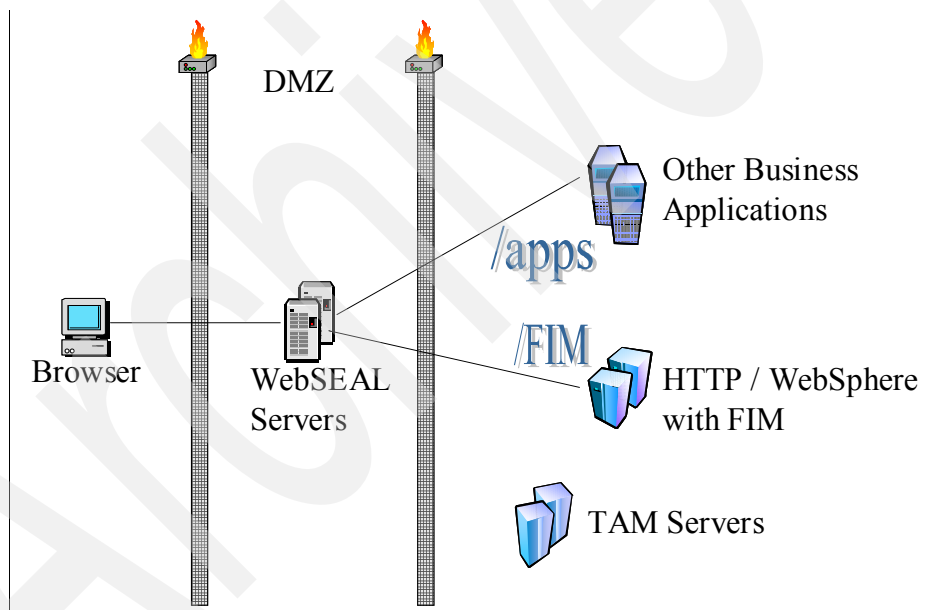


Figure A-1 Using WebSEAL as a point of contact

Figure A-2 on page 365 shows a logical deployment architecture with a Web plug-in. Tivoli Federated Identity Manager runs in the same WebSphere cluster as the business applications, and the WebSphere and Access Manager Web plug-in are installed against the same point of contact Web server. The important thing to note here is that Tivoli Federated Identity Manager shares the same named virtual host (and hence URL name space) as the applications.

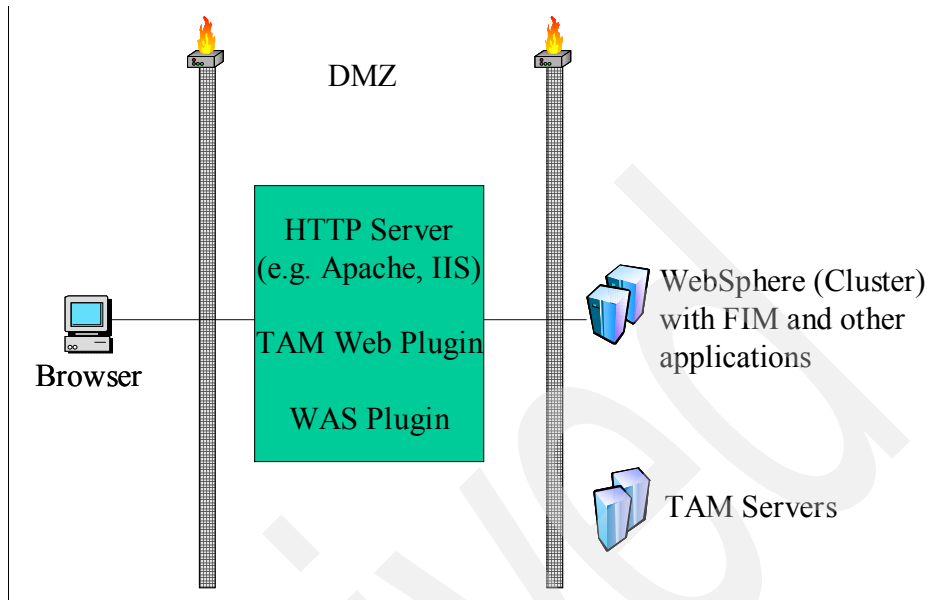


Figure A-2 Web plug-in as a point of contact

For Tivoli Federated Identity Manager's federated user life cycle management, Tivoli Federated Identity Manager plays one or both of two possible roles in federations. These roles are that of an identity provider or service provider. Tivoli Federated Identity Manager's integration requirements are quite different in each of these roles, and the discussion below is split into each role.

Identity provider integration

The following sections describe the necessary steps to configure WebSEAL or Web plug-ins for an identity provider.

As an identity provider, the user is required to authenticate with the point of contact server (WebSEAL or Web plug-in), and Tivoli Federated Identity Manager expects to receive information about that authenticated user when performing federated transactions such as single sign-on.

The integration interfaces for Tivoli Federated Identity Manager in the role of an identity provider are a set of HTTP headers that Tivoli Federated Identity Manager expects to find in all requests. Whether these headers come from the

WebSEAL reverse proxy or the Web plug-ins is irrelevant. The required headers are:

- ▶ `iv-creds` contains the Access Manager credential for the user. Acting as an identity provider, this is how Tivoli Federated Identity Manager determines the current identity of the user.
- ▶ `iv-user` contains the Access Manager user name.
- ▶ `iv_server_name` contains the name of the Access Manager server managing this user's session (used for logout).
- ▶ `user_session_id` contains the identifier of this users Access Manager session (used for logout).

The rest of this section discusses the configuration of these requirements for WebSEAL and the plug-ins.

Configuring WebSEAL as an identity provider

Configuring WebSEAL as an identity provider with Tivoli Federated Identity Manager consists of the following tasks:

- ▶ Updating the WebSEAL configuration file
- ▶ Creating a junction from WebSEAL to Tivoli Federated Identity Manager
- ▶ Optionally including extended attributes from LDAP in the credential (tag/value)

Updating WebSEAL configuration file

Example A-1 indicates the modifications that need to be made to the WebSEAL configuration file for identity provider configuration. For each stanza, locate the corresponding setting and make the changes shown.

Example: A-1 WebSEAL configuration files settings for identity provider

```
[ba]
# for session termination we recommend to not use basic-authentication
ba-auth = none
```

```
[server]
# unsecured http access should be disabled, particularly if you are using
# browser-post style profiles otherwise your assertions may be visible to
# network sniffers
http = no
```

```
[forms]
```

```
# enable forms auth for https since the users have to be able to login somehow
# this is not compulsory, you could use certificates or other authentication
# techniques
forms-auth=https

[session]
# ITFIM requires user session id's to be available on junction
user-session-ids = yes

# we recommend tracking user session id's with cookies. WebSphere cookies
# are needed for other ITFIM capabilities anyway, so why not use them for
# WebSEAL too.
ssl-id-sessions = no
```

Configuring a junction to Tivoli Federated Identity Manager

The junction connecting WebSEAL to Tivoli Federated Identity Manager can be configured via the Access Manager Web Portal Manager. In addition to creating the junction, we need to modify the junction object to send the user session ID as an HTTP header to Tivoli Federated Identity Manager. Figure A-2 shows the command-line `pdadmin` commands necessary to complete these steps. Note the “-c all” argument; this is equivalent to “-c iv_user,iv_user_l,iv_groups,iv_creds”. Use of SSL is optional, though recommended.

Example: A-2 Configuring WebSEAL junction for identity provider

```
pdadmin -a sec_master -p <sec_master password>

pdadmin sec_master> server task <webseal-server-name> create -t ssl -c all -q
/sps/cgi-bin/query_contents -p <TFIM SPS port> -h <TFIM SPS hostname> /ITFIM
pdadmin sec_master> object modify /WebSEAL/<webseal-server>/ITFIM set attribute
HTTP-Tag-Value user_session_id=user_session_id
```

Configuring extended attributes for credentials in WebSEAL

In many cases (typically non-Liberty) you may wish to share attribute information about the user beyond just their user name in the federated single sign-on token. In order to make these attributes available to the Tivoli Federated Identity Manager mapping rules at the identity provider, it makes sense to include them in the original Access Manager credential by reading them from LDAP during user authentication. This can be done with standard extended attribute support in WebSEAL, also known as tag/value support.

The *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 5.1, SC32-1359*, describes how to configure extended attributes in the credential, and downstream them to your business applications as HTTP

headers if necessary. For Tivoli Federated Identity Manager as an identity provider, they need only be inserted in the credential at authentication time, since this will make them available to mapping rules in the Tivoli Federated Identity Manager configuration.

Configuring Web plug-ins as an identity provider

Configuring Web plug-ins as an identity provider with Tivoli Federated Identity Manager consists of the following tasks:

- ▶ Updating the Web plug-in configuration file
- ▶ Optionally including extended attributes from LDAP in the credential (tag/value)

Updating Web plug-in configuration file

Example A-3 indicates the modifications that need to be made to the Web plug-in configuration file. For each stanza, locate the corresponding setting and make the changes shown.

Example: A-3 Web plug-in configuration settings for identity provider

```
[common-modules]
# by default webpi is configured for basic authentication. We should configure
# for forms authentication.
pre-authzn = forms
authentication = forms
post-authzn = forms

# disable basic authentication
#authentication = BA (either remove or comment out this line)
# post-authzn = BA (either remove or comment out this line)

# by default, webpi does not send the required http headers to ITFIM
post-authzn = iv-headers

#enable tag-value support. this is needed for at least the user_session_id
# even if you are not reading other LDAP attributes
post-authzn = tag-value

[iv-headers]
# webpi must be configured to use the iv_server_name header for sending it's
# aznapi server name to ITFIM
server-name-header = iv_server_name
```

```
[pdweb-plugins]
# WebSEAL automatically prefixes credential extended attributes in the
# credential with "tagvalue_", and the Web-plug-in's don't. To make these
# consistent, we recommend using the same prefix tag for Web plug-ins. Also
# you would have to do this anyway if you want to downstream both the
# user_session_id attribute, and other attributes, since the TAM credential
# will be built with a tagvalue_user_session_id.
#
tag-value-prefix = tagvalue_
```

Configuring extended attributes for credentials in Web plug-ins

In many cases (typically non-Liberty) you may wish to share attribute information about the user beyond just their user name in the federated single sign-on token. In order to make these attributes available to the Tivoli Federated Identity Manager mapping rules at the identity provider, it makes sense to include them in the original Access Manager credential by reading them from LDAP during user authentication. This can be done with standard extended attribute support in the Web plug-ins, also known as tag/value support.

The *IBM Tivoli Access Manager for e-business Plug-in for Web Servers Integration Guide Version 5.1, SC32-1365*, describes how to configure tag value support, and downstream credential attributes to your business applications as HTTP headers if necessary. For Tivoli Federated Identity Manager as an identity provider, they need only be inserted in the credential at authentication time, since this will make them available to mapping rules in the Tivoli Federated Identity Manager configuration.

Service provider integration

As a service provider, Tivoli Federated Identity Manager must first be able to process unauthenticated transactions and determine authentication information about the user based on configured trust relationships, then return that authentication information to the point of contact server (WebSEAL or Web plug-in) so that an authenticated session can be established for the user.

The primary integration interface for Tivoli Federated Identity Manager in the role of a service provider is the External Authentication Interface (EAI). This capability is shared by both WebSEAL and the Web plug-ins.

As a service provider, Tivoli Federated Identity Manager also requires the set of HTTP headers described previously for identity provider integration. These are needed for other federated user life cycle management operations such as single

logout. In that regard, the required configuration for a service provider is a super set of that for identity providers.

External Authentication Interface

Tivoli Federated Identity Manager provides an authentication mechanism through its single sign-on protocol service. We are making use of this capability at the service provider side of our identity federations, to allow clients to sign in with credentials generated by another party—the identity provider. By integrating Tivoli Federated Identity Manager with Access Manager, we can treat federated single sign-on as just another Access Manager authentication mechanism. To accomplish this, we are utilizing the Access Manager External Authentication Interface; similar to a Cross-Domain Authentication Service (CDAS), this is an interface for integrating external (viewed from a Access Manager perspective) authentication services with Access Manager. Unlike CDAS, however, for which the integration point is a C-based shared library, EAI uses HTTP for communication with the authentication service. This allows the service to be implemented using any HTTP-capable programming language. In the case of WebSEAL, this can be deployed on a junctioned server (which is exactly what we do with Tivoli Federated Identity Manager). In the case of the Web plug-ins, this can be any URL served by the Web server the plug-in is installed into.

The External Authentication Interface introduces two new concepts to the Access Manager authentication terminology: *Trigger URIs* and *EAI headers*. We will discuss these in turn.

Trigger URIs

The External Authentication Interface is designed to co-exist peacefully with the WebSEAL and Web plug-in internal authentication mechanisms. Because of this, WebSEAL and the plug-ins never enforce or request EAI authentication. Instead, authentication will be triggered when an HTTP response comes from one of the configured Trigger URIs—patterns configured in the [eai-trigger-urls] stanza of the configuration file. The corresponding response is then examined for the presence of EAI headers; if present, a credential is built based upon these, and an Access Manager WebSEAL or Web plug-in session is established.

EAI headers

When authenticating a client through EAI, WebSEAL and the plug-ins play no part in the actual authentication process; this is all delegated to the EAI service. When the EAI service has completed the authentication, it communicates the details of the authenticated principal back via response headers. These come in

two flavors: One type of authentication header contains an Access Manager user ID, and separate discrete attributes used to build a credential; while the other set contains a *Privilege Attribute Certificate (PAC)*, which is converted directly into a credential associated with the user's session. The two approaches are mutually exclusive; if the PAC is present, the discrete attributes will be ignored. Additionally, there is a header common for both cases; this specifies a URL to redirect the client to after authentication. The names of the headers are configured in the [eai] stanza of the WebSEAL or plug-in configuration file; the configuration settings are explained in Table A-2 on page 373.

External Authentication Interface example

This section shows an example of EAI authentication. Table A-1 contains the relevant parameters for an example scenario using WebSEAL.

Table A-1 Example setup using WebSEAL

Configuration variable	Configuration value
WebSEAL hostname	www.example.com
Requested protected page URL	https://www.example.com/secure/index.jsp
Junction for EAI Authentication Service	/loginapp
EAI trigger URI	/loginapp/dologin/*
Custom login page	/loginapp/login.jsp
Login form POST target	/loginapp/dologin/auth

Key points to keep in mind here:

- ▶ The supplied credentials in the example are a user name and a corresponding password in an HTML form, but this is just used as an example. WebSEAL silently passes on the request, so it could contain anything that is appropriate for the receiving application.
- ▶ When a request matches one of the configured trigger URIs, WebSEAL will check the response for EAI headers; if these are absent, the response is proxied to the client as for any other request.
- ▶ The use of trigger URIs instead of *EAI-enabled junctions* gives us more fine-grained control over where to employ this mechanism. This is desirable, not because the response header check is a resource-intensive operation, but because we can limit the URI space from which we will trust authentication information; this allows hosting other (non-EAI) applications on the same application server—essential with the Web plug-ins.

Figure A-3 shows the interaction between WebSEAL and the EAI Authentication Service. The same flows are applicable for use with the Web plug-ins. A detailed description of the steps follows Figure A-3.

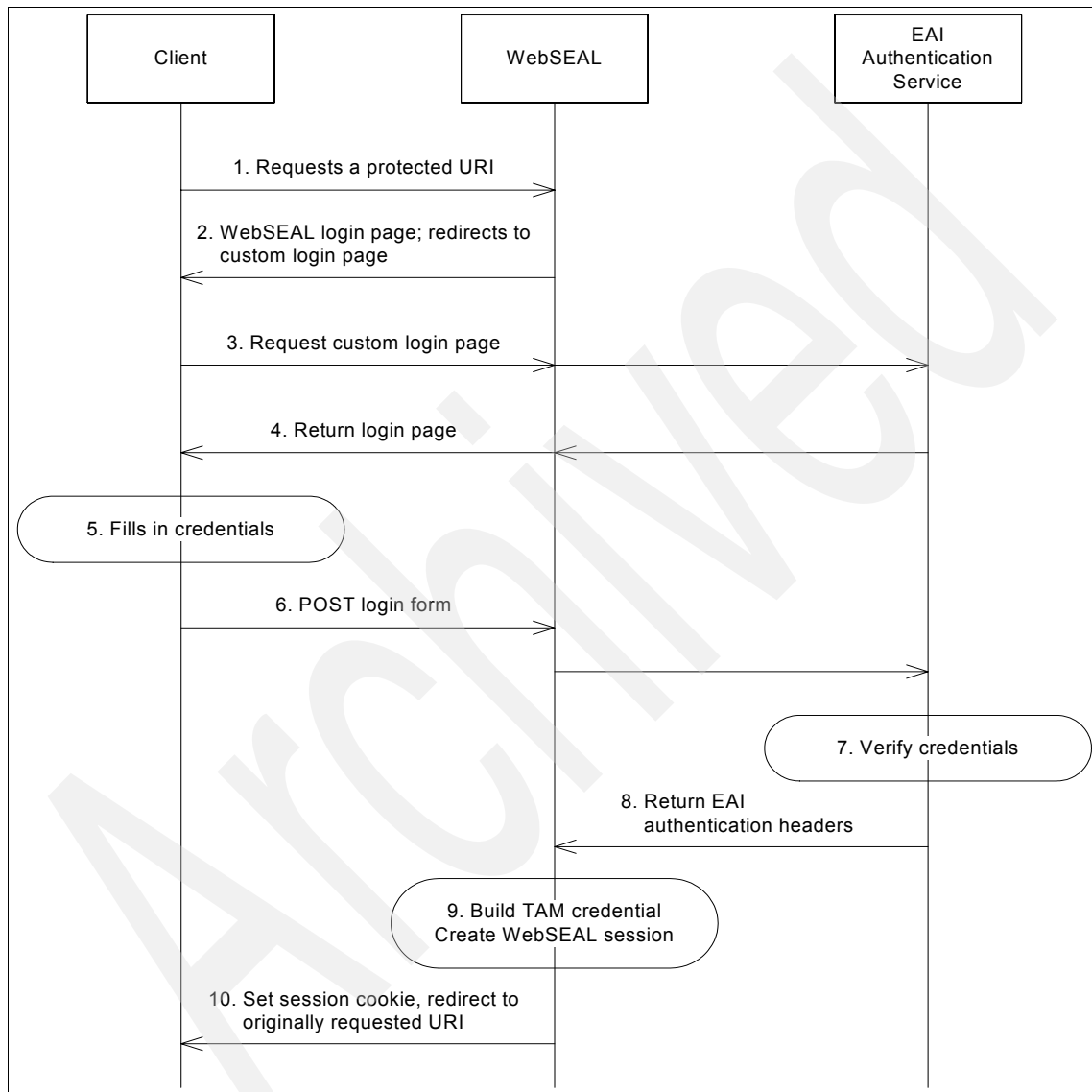


Figure A-3 Interactions between WebSEAL and EAI Authentication Service

Figure A-3 is explained below.

1. An unauthenticated client requests the protected resource.

`https://www.example.com/secure/index.jsp`

2. WebSEAL responds with the login page; this contains an HTTP / 302 redirect to a custom login page, with the originally requested URL as a parameter:

`https://www.example.com/login/login.jsp?url=https://www.example.com/secure/index.jsp`

3. The client requests a custom login page.
4. Response with login page; this contains the originally requested URL as a hidden form field.
5. The client fills a user name and password into the login form.
6. The login form is posted to the EAI Authentication Service; the target URL `https://www.example.com/login/dologin/auth` matches the configured EAI trigger URI `/login/dologin/*`, so WebSEAL will check the response for the presence of EAI headers.
7. The EAI Authentication Service validates the user name and password.
8. The EAI Authentication Service sends response. The response contains the EAI headers for the user ID and redirect URL (the originally requested URL).
9. WebSEAL builds a credential based on contents in the EAI headers, and creates a WebSEAL session for the client.
10. WebSEAL sends a response containing the session cookie and an HTTP / 302 redirect to the originally requested URL:

`https://www.example.com/secure/index.jsp`

EAI header variables reference

Table A-2 describes the configuration file settings used to name EAI headers in WebSEAL, and lists the values used by Tivoli Federated Identity Manager (that is, Tivoli Federated Identity Manager returns headers with these names, so WebSEAL must be configured as shown to recognize them). The values used in this table will be referenced later in this section when we discuss WebSEAL and Web plug-in configuration for EAI. The Web plug-ins use configuration settings that have the same name as WebSEAL, but without the `eai-` prefix. For brevity they are not included in the table.

Table A-2 EAI header names

Config setting	Value used for Tivoli Federated Identity Manager	Description of corresponding header
<code>eai-pac-header</code>	<code>am-fim-eai-pac</code>	Contains a PAC; this takes precedence over user ID.

Config setting	Value used for Tivoli Federated Identity Manager	Description of corresponding header
eai-pac-svc-header	am-fim-eai-pac-svc	Names the service WebSEAL should use to convert the PAC. Optional. The default service is used if not specified.
eai-user-id-header	am-fim-eai-user-id	Specifies the Access Manager user ID of the authenticated user.
eai-auth-level-header	am-fim-eai-auth-level	The authentication level assigned to the client. Optional. Defaults to 1. Corresponds to the Access Manager credential attribute AZN_C_AUTHN_LEVEL.
eai-qop-header	am-fim-eai-qop	The quality of protection. Optional. Corresponds to the Access Manager credential attribute AZN_C_AUTHN_QUALITY and the CDAS input attribute XAUTHN_QOP.
eai-xattrs-header	am-fim-eai-xattrs	A comma-separated list of headers whose contents are added to the Access Manager credential as extended attributes. Optional.
eai-redirect-url-header	am-fim-eai-redirect-url	URL to redirect the client to after successful authentication.

Configuring WebSEAL as a service provider

The following sections outline the configuration requirements for WebSEAL acting as a service provider. This includes EAI configuration, and the requirements for HTTP headers from WebSEAL as for an identity provider.

Configuring WebSEAL as a service provider with Tivoli Federated Identity Manager consists of the following tasks:

- ▶ Updating the WebSEAL configuration file
- ▶ Creating a junction from WebSEAL to Tivoli Federated Identity Manager
- ▶ Applying an Access Manager policy to trigger URLs for EAI
- ▶ Optionally sending credential extended attributes as HTTP headers to business applications (tag/value)

Updating WebSEAL configuration file

Example A-4 indicates the modifications that need to be made to the WebSEAL configuration file. For each stanza, locate the corresponding setting and make the changes shown.

Example: A-4 WebSEAL configuration file settings service provider

```
[ba]
# EAI is incompatible with basic authentication, so this must be disabled:
ba-auth = none

[server]
# unsecured http access should be disabled, particularly if you are using
# browser-post style profiles otherwise your assertions may be visible to
# network sniffers
http = no

[forms]
# enable forms auth for https, particularly for liberty where you are going
# to be doing account linking. The users need to be able to login here locally!
forms-auth=https

[session]
# we recommend tracking user session id's with cookies. WebSphere cookies
# are needed for other ITFIM capabilities anyway, so why not use them for
# WebSEAL too.
ssl-id-sessions = no

# Tivoli Federated Identity Manager SPS needs access to user session ID's:
user-session-ids = yes

[authentication-mechanisms]
# Load the shared library implementing EAI (note - this is for linux platform)
# Similar path and library names exist for other platforms:
ext-auth-interface = /opt/pdwebtrte/lib/libeaiauthn.so

[acct-mgt]
# needed for any errors encountered during EAI authentication
eai-auth-error = eaiautherror.html

[eai]
# allow eai authentication via https only (we are not using http)
eai-auth=https
# settings for eai headers - these are the values used by ITFIM SPS
eai-pac-header = am-fim-eai-pac
eai-pac-svc-header = am-fim-eai-pac-svc
eai-user-id-header = am-fim-eai-user-id
eai-auth-level-header = am-fim-eai-auth-level
```

```
eai-qop-header = am-fim-eai-qop
eai-xattrs-header = am-fim-eai-xattrs
eai-redirect-url-header = am-fim-eai-redirect-url
```

```
[eai-trigger-urls]
# NOTE - these entries will vary, and there should be one entry for each
# federation you have acting as a service provider. The entry should point
# to the login URL for the federation, since this is the URL that will have
# EAI headers returned from it to WebSEAL.
trigger = /ITFIM/sps/samlfed/saml/login
trigger = /ITFIM/sps/myfed2/login/url
```

Configuring a junction to Tivoli Federated Identity Manager

The junction connecting WebSEAL to Tivoli Federated Identity Manager can be configured via the Access Manager Web Portal Manager. In addition to creating the junction, we need to modify the junction object to send the user session ID as an HTTP header to Tivoli Federated Identity Manager. Example A-5 shows the command-line **pdadmin** commands necessary to complete these steps. Note the “-c all” argument; this is equivalent to “-c iv_user,iv_user_l,iv_groups,iv_creds”. Use of SSL is optional, though recommended.

Example: A-5 Configuring WebSEAL junction for service provider

```
pdadmin -a sec_master -p <sec_master password>

pdadmin sec_master> server task <webseal-server-name> create -t ssl -c all -q
/sps/cgi-bin/query_contents -p <TFIM SPS port> -h <TFIM SPS hostname> /ITFIM

pdadmin sec_master> object modify /WebSEAL/<webseal-server>/ITFIM set attribute
HTTP-Tag-Value user_session_id=user_session_id
```

The junction connecting WebSEAL to Tivoli Federated Identity Manager can be configured either via the Access Manager Web Portal Manager or, as shown here, with the **pdadmin** command-line tool. Note the “-c all” argument; this is equivalent to “-c iv_user,iv_user_l,iv_groups,iv_creds”. Use of SSL is optional, though recommended.

```
pdadmin -a sec_master -p <sec_master password>
pdadmin sec_master> server task <webseal-server-name> create -t ssl -c all -p
<TFIM SPS port> -h <TFIM SPS hostname> /ITFIM
```

Access Manager policy for trigger URLs for EAI

The trigger URLs configured for EAI authentication are to permit a user to authenticate to WebSEAL. As such, an unauthenticated user must be able to

access these URLs. Example A-6 shows example `pdadmin` commands for creating an ACL that allows unauthenticated access and attaching it to the URLs shown in the example configuration from Example A-6.

Example: A-6 Attaching unauthenticated ACL to WebSEAL EAI trigger URLs

```
pdadmin -a sec_master -p <sec_master password>
pdadmin sec_master> acl create unauth_ACL
pdadmin sec_master> acl modify unauth_ACL set group iv-admin TcmdbsvaBRrx1
pdadmin sec_master> acl modify unauth_ACL set group webseal-servers Tgmdbsrx1
pdadmin sec_master> acl modify unauth_ACL set user sec_master TcmdbsvaBRrx1
pdadmin sec_master> acl modify unauth_ACL set any-other Trx
pdadmin sec_master> acl modify unauth_ACL set unauthenticated Trx
pdadmin sec_master> acl attach
/WebSEAL/<webseal_server>/ITFIM/sps/samlfed/saml/login unauth_ACL
pdadmin sec_master> acl attach
/WebSEAL/<webseal_server>/ITFIM/sps/myfed2/login/url unauth_ACL
```

Sending extended attributes as HTTP headers with WebSEAL

After performing a federated single sign-on and establishing a session with WebSEAL, it is quite likely that the Access Manager credential built for the user will contain extended attributes that you want to downstream to backend applications. These must be prefixed with `tagvalue_` if they are to be sent as HTTP headers with WebSEAL. This can be done with standard extended attribute support in WebSEAL, also known as `tag/value` support. This allows you to send the extended attributes in the credential as HTTP headers to junctioned applications.

The *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 5.1*, SC32-1359, describes how to configure WebSEAL and junctions for handling extended attributes as HTTP headers.

Configuring Web plug-ins as a service provider

When using Tivoli Federated Identity Manager with the Access Manager Web plug-ins, both the Access Manager Web plug-in and the WebSphere Web plug-in are configured on the same point of contact Web server. Care must be taken to ensure that you are using versions of each plug-in that are supported for the Web server you are configuring them against.

Note: These instructions are only valid for Access Manager 6.0 Web plug-ins and later.

Configuring the Access Manager Web plug-ins as a service provider with Tivoli Federated Identity Manager consists of the following tasks:

- ▶ Updating the plug-in configuration file
- ▶ Applying the Access Manager policy to trigger URLs for EAI
- ▶ Optionally sending credential extended attributes as HTTP headers to business applications (tag/value)

Updating Web plug-in configuration file

Example A-7 indicates the modifications that need to be made to the Web plug-in configuration file. For each stanza, locate the corresponding setting and make the changes shown.

Example: A-7 Web plug-in configuration settings for service provider

```
[common-modules]
# by default webpi is configured for basic authentication. We should configure
# for forms authentication.
pre-authzn = forms
authentication = forms
post-authzn = forms

#include EAI authentication
pre-authzn = ext-auth-int
response = ext-auth-int
authentication = ext-auth-int
post-authzn = ext-auth-int

# disable basic authentication
#authentication = BA (either remove or comment out this line)
#post-authzn = BA (either remove or comment out this line)

# by default, webpi does not send the required http headers to ITFIM
post-authzn = iv-headers

#enable tag-value support. this is needed for at least the user_session_id
post-authzn = tag-value

[iv-headers]
# webpi must be configured to use the iv_server_name header for sending it's
# aznapi server name to ITFIM
server-name-header = iv_server_name

#
# Settings for EAI.
#
```



```

[modules]
ext-auth-int = pdwpi-ext-auth-int-module

[authentication-mechanisms]
# this is shown for linux, similar paths exist for other platforms
ext-auth-interface = /opt/pdwebvrte/lib/libeiaiauthn.so

[ext-auth-int]
# actually for this configuration the auth-url doesn't matter because
# we will prompt for forms login first. If EAI was the only authentication
# mechanism enabled then this is the page you would be redirected to when
# accessing a protected resource
auth-url = /some/login/url
trigger-url = /sps/samlfed/saml/login
trigger-url = /sps/myfed2/login/url

redirect-url-hdr-name = am-fim-eai-redir-url
pac-hdr-name = am-fi-eai-pac
pac-svc-id-hdr-name = am-fim-eai-pac-svc
user-id-hdr-name = am-fim-eai-user-id
user-auth-level-hdr-name = am-fim-eai-auth-level
user-qop-hdr-name = am-fim-eai-qop
user-ext-attr-list-hdr-name = am-fim-eai-xattrs

#
# Finally, webpi's default configuration for the size of the buffers used to
# transfer data between the web server and the authorization server is too
# small for EAI to function properly. It is usually necessary to increase this
# parameter from 10000 to 50000 bytes.
[proxy-if]
worker-size = 50000

[pdweb-plugins]
# WebSEAL automatically prefixes credential extended attributes in the
# credential with "tagvalue_", and the Web-plug-in's don't. To make these
# consistent, we recommend using the same prefix tag for Web plug-ins. Also
# you would have to do this anyway if you want to downstream both the
# user_session_id attribute, and other attributes, since the TAM credential
# will be built with a tagvalue_user_session_id.
#
tag-value-prefix = tagvalue_

```

Access Manager policy for trigger URLs

The trigger URLs configured for EAI authentication are to permit a user to authenticate to Web plug-ins. As such, an unauthenticated user must be able to access these URLs. Example A-8 shows example `pdadmin` commands for creating an ACL, which allows unauthenticated access and attaching it to the URLs shown in the example configuration from Example A-8.

Example: A-8 Attaching unauthenticated ACL to WebSEAL EAI trigger URLs

```
pdadmin -a sec_master -p <sec_master password>
pdadmin sec_master> acl create unauth_webpi
pdadmin sec_master> acl modify unauth_webpi set group iv-admin TcmdbsvaBRrx1
pdadmin sec_master> acl modify unauth_webpi set group webseal-servers Tgmdbsrx1
pdadmin sec_master> acl modify unauth_webpi set user sec_master
TcmdbsvaBRrx1[PDWebPI]r
pdadmin sec_master> acl modify unauth_webpi set unauthenticated Trxr
pdadmin sec_master> acl modify unauth_webpi set any-other Trx[PDWebPI]r
pdadmin sec_master> acl attach
/PDWebPI/<virtual_hostname>/sps/samlfed/saml/login unauth_webpi
pdadmin sec_master> acl attach /PDWebPI/<virtual_hostname>/sps/myfed2/login/url
unauth_webpi
```

Sending extended attributes as HTTP headers with Web plug-ins

After performing a federated single sign-on and establishing a session with Web plug-ins, it is quite likely that the Access Manager credential built for the user will contain extended attributes that you want to downstream to backend applications. This can be done with standard tag/value support in the Web plug-ins. This allows you to send the extended attributes in the credential as HTTP headers to junctioned applications.

The *IBM Tivoli Access Manager for e-business Plug-in for Web Servers Integration Guide Version 5.1*, SC32-1365, describes how to configure tag value support.



Identity mapping rules

This appendix describes an approach to authoring the XSL identity mapping rules for Tivoli Federated Identity Manager, and also contains all of the identity mapping rules used in the scenarios in this book.

Some of the identity mapping rules used in these scenarios call out to Java code, and the sample Java code is also presented in this appendix.

Authoring identity mapping rules

One of the most powerful and differentiating features of Tivoli Federated Identity Manager is the ability to implement rich identity mapping capabilities between different token formats using the XML Stylesheet Language (XSL). XSL is a transformation language that allows you to use templates to transform XML documents from one format to another. For a good introduction to XSL, try the tutorial at:

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

The *IBM Tivoli Federated Identity Manager Administration Guide Version 6.0*, GC32-1668-00, contains useful base information on mapping rules, and points to the example mapping rules shipped with Tivoli Federated Identity Manager. This appendix expands upon this documentation with some techniques for authoring your own rules. We also describe what is required for calling your own Java code from the mapping rules. This is a common requirement in real-world use cases, and was used in two of the scenarios described in this book.

XSL mapping rules are required whenever tokens are processed at the Tivoli Federated Identity Manager trust service, and provide you with the opportunity to modify or completely change the user name, groups (if applicable to the token type), and extended attributes associated with the resulting token.

Consider, for example, a SAML single sign-on scenario:

- ▶ At the identity provider the source token type will be a Tivoli Access Manager credential (originally provided to the Tivoli Federated Identity Manager single sign-on protocol service by WebSEAL in the iv-creds HTTP header), and the destination token type is the SAML assertion that is used in the SAML protocol to sign-on to the service provider.
- ▶ At the service provider the source token type will be the SAML assertion and the destination token will be an Access Manager credential used to perform an EAI login to WebSEAL.

The Tivoli Federated Identity Manager trust service includes token modules for different token types. These modules validate and transform the source token type into an internal XML representation of a credential called an STSUniversalUser. They also have the ability to take an STSUniversalUser and transform it into a token of their own type. Understanding the format of an STSUniversalUser, and how different Tivoli Federated Identity Manager-supported token types are mapped to and from the STSUniversalUser, is the key to authoring mapping rules.

The rest of this section describes:

- ▶ The STSUniversalUser schema

- ▶ How token types are mapped between their native format and the STSUniversalUser format
- ▶ Calling Java code from XSL mapping rules
- ▶ Developer tricks for authoring and testing mapping rules

STSUniversalUser schema

Example B-1 shows the XML Schema for the STSUniversalUser. This information, when combined with the description in the *IBM Tivoli Federated Identity Manager Administration Guide Version 6.0, GC32-1668-00*, will help provide a complete understanding of what an STSUniversalUser looks like.

At a minimum, be aware that the STSUniversalUser is divided into three groups of attributes: Principal, Groups, and AttributeList, and each of these contains name/value pairs of information about the user.

Example: B-1 STSUniversalUser XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:ibm:names:ITFIM:1.0:stsuser"
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser"
  elementFormDefault="qualified">

  <xsd:element name="STSUniversalUser">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Principal"
          type="stsuser:PrincipalType" minOccurs="1" maxOccurs="1" />
        <xsd:element name="GroupList"
          type="stsuser:GroupListType" minOccurs="0" maxOccurs="1" />
        <xsd:element name="AttributeList"
          type="stsuser:AttributeListType" minOccurs="0" maxOccurs="1"
        />
      </xsd:sequence>
      <xsd:attribute name="version" type="xsd:string"
        use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="PrincipalType">
    <xsd:sequence>
      <xsd:element name="Attribute" type="stsuser:AttributeType"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
```

```

<xsd:complexType name="AttributeType">
  <xsd:sequence>
    <xsd:element name="Value" type="xsd:string" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="xsd:string" use="optional" />
</xsd:complexType>

<xsd:complexType name="AttributeListType">
  <xsd:sequence>
    <xsd:element name="Attribute" type="stsuser:AttributeType"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="GroupListType">
  <xsd:sequence>
    <xsd:element name="Group" type="stsuser:GroupType"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="GroupType">
  <xsd:sequence>
    <xsd:element name="Attribute" type="stsuser:AttributeType"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="xsd:string" use="optional" />
</xsd:complexType>

</xsd:schema>

```

Mapping between STSUniversalUser and native tokens

Before writing XSL mapping rules you must understand how the token type you are transforming from will look as an STSUniversalUser. You must also understand the requirements on the resulting STSUniversalUser so that the token module in the Tivoli Federated Identity Manager trust service will successfully issue the resulting token.

For each of the following token types, many of which are used in the scenarios in this book, we will show an example STSUniversalUser (useful if this is your starting token type), and describe the requirements and options on the

STSTUniversalUser for issuing a token (useful if this is your destination token type):

- ▶ Tivoli Access Manager credential
- ▶ SAML 1.0 token
- ▶ SAML 1.1 token
- ▶ Liberty 1.1 token
- ▶ Liberty 1.2 token
- ▶ UsernameToken

In addition to the above token types, Tivoli Federated Identity Manager offers an API to implement your own token type. It is also likely that Tivoli Federated Identity Manager will support additional token types in future releases. In any case, the pattern remains the same—what is required is an understanding of the STSTUniversalUser format of the starting token, and the requirements on the STSTUniversalUser for issuing a token of the destination token type.

Tivoli Access Manager credential

This section details:

- ▶ The STSTUniversalUser format generated by the Tivoli Federated Identity Manager trust service after validating an Access Manager credential
- ▶ Requirements for the STSTUniversalUser so that the Tivoli Federated Identity Manager trust service will issue an Access Manager credential

STSTUniversalUser for Access Manager credential

Example B-2 shows an example STSTUniversalUser as generated by the Tivoli Federated Identity Manager trust service when validating an Access Manager credential. This scenario typically occurs at an identity provider during a federated single sign-on operation, and also occurs in Chapter 10, “Use case 4 - Web services security management” on page 291, when our Access Manager credential is exchanged at the client-side Web services gateway for a SAML assertion.

Example: B-2 Sample STSTUniversalUser for Access Manager credential

```
<stsuser:STSTUniversalUser xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
  <stsuser:Principal>
    <stsuser:Attribute name="uuid"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>9d5f1ea8-df36-11d9-872b-000c29a951ea</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="domain"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>Default</stsuser:Value>
    </stsuser:Attribute>
```

```

    <stsuser:Attribute name="name"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>emp1</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="registryid"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>CN=Employee
One,CN=Users,DC=bigcorp,DC=com</stsuser:Value>
    </stsuser:Attribute>
</stsuser:Principal>
<GroupList xmlns="urn:ibm:names:ITFIM:1.0:stsuser">
    <stsuser:Group name="employees"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Attribute name="uuid"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
<stsuser:Value>6f0f791c-ea49-11d9-a4eb-000c29d2099e</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="registryid"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
<stsuser:Value>CN=employees,CN=Users,DC=bigcorp,DC=com</stsuser:Value>
    </stsuser:Attribute>
    </stsuser:Group>
</GroupList>
<stsuser:AttributeList>
    <stsuser:Attribute name="AZN_CRED_AUTH_METHOD"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>kerberosv5</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_BROWSER_INFO"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
SV1)</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_PRINCIPAL_NAME"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>emp1</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="tagvalue_activedir_cn"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>Employee One</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AUTHENTICATION_LEVEL"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>0</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_PRINCIPAL_UUID"
type="urn:ibm:names:ITFIM:5.1:accessmanager">

```



```

        <stsuser:Value>9d5f1ea8-df36-11d9-872b-000c29a951ea</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_GROUPS"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>employees</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_PRINCIPAL_DOMAIN"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>Default</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_AUTHZN_ID"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>emp1</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_QOP_INFO"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>None</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_IP_ADDRESS"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>0x03050309</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_GROUP_REGISTRY_IDS"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
<stsuser:Value>CN=employees,CN=Users,DC=bigcorp,DC=com</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_VERSION"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>0x00000510</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_GROUP_UUIDS"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>6f0f791c-ea49-11d9-a4eb-000c29d2099e</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_MECH_ID"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>IV_URAF_V3.0</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AZN_CRED_REGISTRY_ID"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>CN=Employee
One,CN=Users,DC=bigcorp,DC=com</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="tagvalue_activedir_mail"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
        <stsuser:Value>emp1@bigcorp.com</stsuser:Value>
    </stsuser:Attribute>

```

```
<stsuser:Attribute name="AZN_CRED_AUTHNMECH_INFO"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
  <stsuser:Value>GSS Authentication</stsuser:Value>
</stsuser:Attribute>
</stsuser:AttributeList>
</stsuser:STSTUniversalUser>
```

Issuing an Access Manager credential

This section details the requirements for an STSUniversalUser for issuing an Access Manager credential. If the STSUniversalUser resulting from your XSL mapping rule does not meet these requirements, the Tivoli Federated Identity Manager trust service will fail to issue an Access Manager credential.

The STSUniversalUser *must* have a Principal attribute called name with type urn:ibm:names:ITFIM:5.1:accessmanager. All other Principal attributes will be ignored.

The STSUniversalUser *should* contain an extended attribute for the AUTHENTICATION_LEVEL parameter, with type urn:ibm:names:ITFIM:5.1:accessmanager. Its value must be a number representing a valid Access Manager authentication level. This parameter is necessary for reauthentication to work at the service provider.

The STSUniversalUser *should* contain an extended attribute for AZN_CRED_AUTH_METHOD with type urn:ibm:names:ITFIM:5.1:accessmanager. This is not absolutely required, and is not used in our use case examples, but should be included for consistency since it is carried as an attribute in Access Manager failover cookies, and it is desirable for credentials built during failover to carry the same attributes as was on the originating server.

The STSUniversalUser *may* contain the following standard Access Manager attributes, though at the time of writing their use will generally not affect the operation of Access Manager unless you have Access Manager authorization rules or POPs acting on their values:

- ▶ AZN_CRED_AUTHNMECH_INFO
- ▶ AZN_CRED_BROWSER_INFO
- ▶ AZN_CRED_IP_ADDRESS
- ▶ AZN_CRED_QOP_INFO
- ▶ AZN_CRED_USER_INFO (this is included in audit log, if present)

The STSUniversalUser *may* also contain other extra Group and AttributeList attributes.

The token module can be configured to permit groups to be added to the base Access Manager credential. If enabled, and group names are being included, you must use the type attribute `urn:ibm:names:ITFIM:5.1:accessmanager`.

All other `AttributeList` attributes are typically just appended to the built Access Manager credential; however, the module can be configured to filter these based on type. By default, the type filter is `*`, which will include all attribute types.

SAML 1.0 token

This section details:

- ▶ The `STSEntityUser` format generated by the Tivoli Federated Identity Manager trust service after validating a SAML 1.0 token
- ▶ Requirements on the `STSEntityUser` so that the Tivoli Federated Identity Manager trust service will issue a SAML 1.0 token

STSEntityUser for SAML 1.0 tokens

Example B-3 shows an example `STSEntityUser` as generated by the Tivoli Federated Identity Manager trust service after validating a SAML 1.0 assertion. This scenario typically occurs at a service provider during a federated single sign-on operation, and also occurs in our sample use case 4 at `RBStocks` when the signed SAML assertion sent from `RBTelco` is exchanged for a “local” SAML assertion.

Example: B-3 Sample STSEntityUser for SAML 1.0 assertion

```
<?xml version="1.0" encoding="UTF-8"?>
<stsuser:STSEntityUser
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
  <stsuser:Principal>
    <stsuser:Attribute name="name"
      type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">
      <stsuser:Value>emp1@bigcorp.com</stsuser:Value>
    </stsuser:Attribute>
  </stsuser:Principal>
  <stsuser:AttributeList>
    <stsuser:Attribute name="IssueInstant"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T19:12:35Z</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AuthenticationMethod"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        urn:oasis:names:tc:SAML:1.0:am:password
      </stsuser:Value>
    </stsuser:Attribute>
  </stsuser:AttributeList>
</stsuser:STSEntityUser>
```

```

<stsuser:Attribute name="NotBefore"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>2005-07-05T19:02:35Z</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="MinorVersion"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>0</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="MajorVersion"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>1</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="Issuer"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>https://www.bigcorp.com</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="cn"
  type="http://www.bigcorp.com/cn">
  <stsuser:Value>Employee One</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="AuthenticationInstant"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>2005-07-05T19:12:35Z</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="NotOnOrAfter"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>2005-07-05T19:22:35Z</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="AssertionID"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>
    Assertion-uuide869f5f7-0104-e43d-cff3-8753ba4ddf37
  </stsuser:Value>
</stsuser:Attribute>
</stsuser:AttributeList>
</stsuser:STSEntityUser>

```

Issuing a SAML 1.0 assertion

This section details the requirements on an STSEntityUser for issuing a SAML 1.0 assertion. If the STSEntityUser resulting from your XSL mapping rule does not meet these requirements, the Tivoli Federated Identity Manager trust service will fail to issue the assertion.

The Principal *must* contain a name attribute, and its type must be one of the following supported SAML 1.0 subject types:

- ▶ #emailAddress

- ▶ #X509SubjectName
- ▶ #WindowsDomainQualifiedName
- ▶ urn:oasis:names:tc:SAML:1.0:assertion#emailAddress
- ▶ urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName
- ▶ urn:oasis:names:tc:SAML:1.0:assertion#WindowsDomainQualifiedName

The actual value of the name can be any string value.

An AuthenticationMethod *must* be provided as an extended attribute in the AttributeList with a name of AuthenticationMethod and a type of urn:oasis:names:tc:SAML:1.0:assertion. The value of the AuthenticationMethod *should* be one of the following supported methods, for example, urn:oasis:names:tc:SAML:1.0:am:password.

For a full list of the recommended authentication method URIs, see section 7.1 of the *Assertions and Protocol for the OASIS Security Assertion Markup Language*. This document is available for download from:

<http://www.oasis-open.org/specs/index.php#samlv1.0>

The issuer of the SAML assertion typically comes from the provider ID you configured when you created the federation or partner in the first place. You can, however, override the issuer by including an attribute in the Principal section called issuer with a type of urn:oasis:names:tc:SAML:1.0:assertion.

A SAML NameQualifier used in the Subject can optionally be provided as an extended attribute in the AttributeList with a name of NameQualifier and type of urn:oasis:names:tc:SAML:1.0:assertion.

Other structured elements of the SAML assertion are either optional and not supported or only ever filled in by the Tivoli Federated Identity Manager runtime, possibly influenced by your federation configuration. A good example of this is the SubjectConfirmationMethod, which is always set by Tivoli Federated Identity Manager depending on whether a browser artifact or browser post profile is being used. Similarly, NotBefore and NotOnOrAfter conditions are influenced by federation configuration of the validity period of the assertion.

All other AttributeList attributes are added to a SAML AttributeStatement within the assertion. The module can be configured to filter these based on type. By default the type filter is *, which will include all attribute types.

SAML 1.1 token

This section details:

- ▶ The STSUniversalUser format generated by the Tivoli Federated Identity Manager trust service after validating a SAML 1.1 token
- ▶ Requirements on the STSUniversalUser so that the Tivoli Federated Identity Manager trust service will issue a SAML 1.1 token

STUniversalUser for SAML 1.1 tokens

Example B-4 shows an example STSUniversalUser as generated by the Tivoli Federated Identity Manager trust service after validating a SAML 1.1 assertion. This scenario typically occurs at a service provider during a federated single sign-on operation for the WS-Federation passive requester profile. The mapping rule at RBTelco in use case 2 processes this type of token.

Example: B-4 Sample STSUniversalUser for SAML 1.1 assertion

```
<stsuser:STSUniversalUser
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
  <stsuser:Principal>
    <stsuser:Attribute name="name"
      type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
      <stsuser:Value>emp1@bigcorp.com</stsuser:Value>
    </stsuser:Attribute>
  </stsuser:Principal>
  <stsuser:AttributeList>
    <stsuser:Attribute name="IssueInstant"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T19:32:44Z</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AuthenticationMethod"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        urn:oasis:names:tc:SAML:1.0:am:password
      </stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="NotBefore"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T19:22:44Z</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="MinorVersion"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>1</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="MajorVersion"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>1</stsuser:Value>
```

```

</stsuser:Attribute>
<stsuser:Attribute name="Issuer"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>
    https://www.bigcorp.com/ITFIM/sps/wsfed/wsf
  </stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="cn"
  type="http://www.bigcorp.com/cn">
  <stsuser:Value>Employee One</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="AuthenticationInstant"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>2005-07-05T19:32:44Z</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="NotOnOrAfter"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>2005-07-05T19:42:44Z</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="AssertionID"
  type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuser:Value>
    Assertion-uuide87c677a-0104-e2b5-7623-8753ba4ddf37
  </stsuser:Value>
</stsuser:Attribute>
</stsuser:AttributeList>
</stsuser:STSUniversalUser>

```

Issuing a SAML 1.1 assertion

This section details the requirements for an STSUniversalUser for issuing a SAML 1.1 assertion. If the STSUniversalUser resulting from your XSL mapping rule does not meet these requirements, the Tivoli Federated Identity Manager trust service will fail to issue the assertion.

The Principal *must* contain a name attribute, and its type must be one of the following supported SAML 1.1 subject types:

- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

The actual value of the name can be any string value.

An AuthenticationMethod *must* be provided as an extended attribute in the AttributeList with a name of AuthenticationMethod and a type of urn:oasis:names:tc:SAML:1.0:assertion. The value of the AuthenticationMethod *should* be one of the supported methods such as

urn:oasis:names:tc:SAML:1.0:am:password. For a full list of the recommended authentication method URIs, consult the SAML 1.1 specifications.

The issuer of the SAML assertion typically comes from the provider ID you configured when you created the federation or partner in the first place. You can, however, override the issuer by including an attribute in the Principal section called issuer with a type of urn:oasis:names:tc:SAML:1.0:assertion.

A SAML NameQualifier used in the Subject can optionally be provided as an extended attribute in the AttributeList with a name of NameQualifier and type of urn:oasis:names:tc:SAML:1.0:assertion.

Other structured elements of the SAML assertion are either optional and not supported or only ever filled in by the Tivoli Federated Identity Manager Runtime, possibly influenced by your federation configuration. A good example of this is the SubjectConfirmationMethod, which is always set by Tivoli Federated Identity Manager depending on whether a browser artifact or browser post profile is being used. Similarly, NotBefore and NotOnOrAfter conditions are influenced by federation configuration of the validity period of the assertion.

All other AttributeList attributes are added to a SAML AttributeStatement within the assertion. The module can be configured to filter these based on type. By default the type filter is *, which will include all attribute types.

Liberty 1.1 token

This section details:

- ▶ The STSUniversalUser format generated by the Tivoli Federated Identity Manager trust service after validating a Liberty 1.1 token
- ▶ Requirements for the STSUniversalUser so that the Tivoli Federated Identity Manager trust service will issue a Liberty 1.1 token

STSUniversalUser for Liberty 1.1 tokens

This scenario typically occurs at a service provider during a Liberty 1.1 federated single sign-on operation. The Liberty 1.1 STSUniversalUser looks exactly the same as a Liberty 1.2 universal user, so a separate example is not provided.

B-5 shows an example STSUniversalUser as generated by the Tivoli Federated Identity Manager trust service after validating a Liberty 1.2 assertion.

Issuing a Liberty 1.1 assertion

This section details the requirements on an STSUniversalUser for issuing a Liberty 1.1 assertion. If the STSUniversalUser resulting from your XSL mapping

rule does not meet these requirements, the Tivoli Federated Identity Manager trust service will fail to issue the assertion.

The Principal *must* contain a name attribute, and its type must be one of the following supported SAML 1.1 subject types:

- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

The actual value of the name can be any string value.

An AuthenticationMethod *must* be provided as an extended attribute in the AttributeList with a name of AuthenticationMethod and a type of urn:oasis:names:tc:SAML:1.0:assertion. The value of the AuthenticationMethod *should* be a supported SAML 1.0 authentication method such as urn:oasis:names:tc:SAML:1.0:am:password. Please see the SAML and Liberty specifications for details of the recommended authentication URIs.

Other structured elements of the assertion are either optional and not supported or only ever filled in by the Tivoli Federated Identity Manager Runtime, possibly influenced by your federation configuration. A good example of this is the SubjectConfirmationMethod, which is always set by Tivoli Federated Identity Manager depending on whether a browser artifact or browser post profile is being used. Similarly, NotBefore and NotOnOrAfter conditions are influenced by federation configuration.

All other AttributeList attributes are added to a SAML AttributeStatement within the assertion. The module can be configured to filter these based on type. By default the type filter is *, which will include all attribute types. Care should be taken when doing this with Liberty assertions since field experience has shown the some other vendors products do not interoperate with Liberty assertions that contain an AttributeStatement.

Liberty 1.2 token

This section details:

- ▶ The STSUniversalUser format generated by the Tivoli Federated Identity Manager trust service after validating a Liberty 1.2 token
- ▶ Requirements for the STSUniversalUser so that the Tivoli Federated Identity Manager trust service will issue a Liberty 1.2 token

STUniversalUser for Liberty 1.2 tokens

B-5 shows an example STSUniversalUser as generated by the Tivoli Federated Identity Manager trust service after validating a Liberty 1.2 assertion. This

scenario typically occurs at a service provider during a Liberty 1.2 federated single sign-on operation.

Example: B-5 Sample STSUniversalUser for Liberty 1.2 assertion

```
<stsuser:STSUniversalUser
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
  <stsuser:Principal>
    <stsuser:Attribute name="name"
      type="urn:liberty:iff:nameid:federated">
      <stsuser:Value>rbtickets1</stsuser:Value>
    </stsuser:Attribute>
  </stsuser:Principal>
  <stsuser:AttributeList>
    <stsuser:Attribute name="IssueInstant"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T20:07:44Z</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AuthenticationMethod"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        urn:oasis:names:tc:SAML:1.0:am:password
      </stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="NotBefore"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T20:06:44Z</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AudienceRestrictionCondition.Audience"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        https://www.rbtickets.com/ITFIM/sps/liberty12/liberty
      </stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="issuer"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        https://www.rbtelco.com/ITFIM/sps/libertyfed/liberty
      </stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="AuthenticationInstant"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T20:07:44Z</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="NotOnOrAfter"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>2005-07-05T20:09:44Z</stsuser:Value>
    </stsuser:Attribute>
  </stsuser:AttributeList>
```

Issuing a Liberty 1.2 assertion

This section details the requirements on an STSUniversalUser for issuing a Liberty 1.2 assertion. If the STSUniversalUser resulting from your XSL mapping rule does not meet these requirements, the Tivoli Federated Identity Manager trust service will fail to issue the assertion.

The Principal *must* contain a name attribute, and its type must be one of the following supported SAML 1.1 subject types:

- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- ▶ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

The actual value of the name can be any string value.

An AuthenticationMethod *must* be provided as an extended attribute in the AttributeList with a name of AuthenticationMethod and a type of urn:oasis:names:tc:SAML:1.0:assertion. The value of the AuthenticationMethod *should* be a supported SAML 1.0 authentication method such as urn:oasis:names:tc:SAML:1.0:am:password. Please see the SAML and Liberty specifications for details of the recommended authentication URIs.

Other structured elements of the assertion are either optional and not supported or only ever filled in by the Tivoli Federated Identity Manager runtime, possibly influenced by your federation configuration. A good example of this is the SubjectConfirmationMethod, which is always set by Tivoli Federated Identity Manager depending on whether a browser artifact or browser post profile is being used. Similarly, NotBefore and NotOnOrAfter conditions are influenced by federation configuration.

All other AttributeList attributes are added to a SAML AttributeStatement within the assertion. The module can be configured to filter these based on type. By default the type filter is *, which will include all attribute types. Care should be taken when doing this with Liberty assertions since field experience has shown that some other vendors' products do not interoperate with Liberty assertions that contain an AttributeStatement.

UsernameToken token

This section details:

- ▶ The STSUniversalUser format generated by the Tivoli Federated Identity Manager trust service after validating a UsernameToken

- Requirements for the STSUniversalUser so that the Tivoli Federated Identity Manager trust service will issue a UsernameToken

The UsernameToken format supported by Tivoli Federated Identity Manager is the wss:UsernameToken profile described in the document at:

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

STSUniversalUser for UsernameToken

Example B-6 shows an example STSUniversalUser as generated by the Tivoli Federated Identity Manager trust service after validating a UsernameToken. This scenario typically occurs as part of a Web Service Security Management operation, such as the EchoApplication shipped as an example with Tivoli Federated Identity Manager.

Example: B-6 Sample STSUniversalUser for UsernameToken

```
<stsuser:STSUniversalUser
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
  <stsuser:Principal>
    <stsuser:Attribute name="Username"

type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <stsuser:Value>wasadmin</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="name"

type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <stsuser:Value>wasadmin</stsuser:Value>
    </stsuser:Attribute>
    <stsuser:Attribute name="Password"

type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">
      <stsuser:Value>*****</stsuser:Value>
    </stsuser:Attribute>
  </stsuser:Principal>
</stsuser:STSUniversalUser>
```

Issuing a UsernameToken

This section details the requirements on an STSUniversalUser for issuing a UsernameToken. If the STSUniversalUser resulting from your XSL mapping rule does not meet these requirements, the Tivoli Federated Identity Manager trust service will fail to issue the token.

The Principal *must* contain an attribute carrying the user name, and it must be called either Username or name with a type of <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd>.

The value of this attribute is just the user's name.

The Principal can optionally contain a Password attribute. If a type is supplied for the Password, that type will be included in the constructed token; otherwise, the default value of <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText> will be used, and the password will be treated as a cleartext password. Cleartext passwords can be handled one of two ways, depending on how the token module is configured. Either the cleartext password is included without modification, or the token module can be configured to compute a password digest for you and include that in the resulting token.

You can also pre-set the type attribute of your password to <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest> for a password digest you have computed yourself. Password digests presented in this fashion are included without modification in the resulting token.

The AttributeList of the STSUniversalUser can optionally contain an attribute called Created with a type of <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>, which should contain a preformatted time value, which is the time the user name token is created. If this attribute is found, and the token module is configured to include a created timestamp, it is added as-is to the resulting token. If this attribute is not found and the module is configured to add a timestamp, the token module will add one based on the current time.

By way of configuration, the UsernameToken module can also add Nonce and Timestamp elements to the token. No attributes are required in the STSUniversalUser for these elements.

Calling Java code from mapping rules

The Tivoli Federated Identity Manager XSLT processor is based in Java, and it is a simple matter to write your own Java code and call it from your XSL mapping rule. Two of the scenarios in this book have examples of this. The first is in use case 1 at RBTravel where Java code is used to poll for the existence of, and create if necessary, the user who is trying to single sign-on via SAML 1.0. The

second example is use case 4 at RBStocks where Java code is used to check the inbound user's e-mail address against a text file blacklist.

Learning how to call Java from XSL

The simplest way to become familiar with calling Java from XSL is to take one of the examples from this book and modify it to suit your own purposes. One of the more useful online references for learning about calling other languages from XSL is:

<http://xml.apache.org/xalan-j/extensions.html#format-date-stylesheet>

Distributing Java code

Once you have written and compiled your Java code and configured your XSL rule to call it, you need to make the compiled classes or jar file available to the classpath of the JVM executing the XSL. This can be done several ways. The quick and dirty way, particularly useful during development, is to drop your jar file into the WebSphere/AppServer/classes directory. A more distributable approach (from a cluster point of view) is to distribute your jar to all nodes in the same location and then use a WebSphere shared library to include it in the classpath.

Developer tricks for mapping rules

This section outlines a few techniques for developers that may be useful in the development of mapping rules.

Working with Access Manager credentials

When developing mapping rules that map to or from Access Manager credentials, one of the most valuable resources for understanding what is in the original or final Access Manager credential is the WebSEAL epac demo program that ships with the WebSEAL pdwebtrt. There is a readme included that demonstrates how to set it up. When successfully authenticated to WebSEAL, accessing this epac CGI program with a browser will show you a screen similar to that in Figure B-1 on page 401.

The epac CGI essentially unpacks the Access Manager credential and shows all of the attributes it contains. These map directly to elements you will find in the STSUniversalUser. This is particularly useful when working with tag-value extended attributes, or other non-obvious attributes in the credential.

Tivoli Access Manager EPAC CGI - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address https://www.bigcorp.com/cgi-bin/epac

Links BigCorp RBTelco epac RBTravel epac WSFed

epac:
 BAKs3DCCBFYMADCCBFAwggRMAgIFEDBYMCYwHgIEnV8eqAIDAN82AgIR2QICAIcCASsEBgAMKaf
 epac size: 1492
 Number of creds: 1

Index	Name	Value(s)
0	AUTHENTICATION_LEVEL	[0]: 0
1	AZN_CRED_AUTHNMECH_INFO	[0]: GSS Authentication
2	AZN_CRED_AUTHZN_ID	[0]: emp1
3	AZN_CRED_AUTH_METHOD	[0]: kerberosv5
4	AZN_CRED_BROWSER_INFO	[0]: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
5	AZN_CRED_GROUPS	[0]: employees
6	AZN_CRED_GROUP_REGISTRY_IDS	[0]: CN=employees,CN=Users,DC=bigcorp,DC=com
7	AZN_CRED_GROUP_UUIDS	[0]: 6f0f791c-ea49-11d9-a4eb-000c29d2099e
8	AZN_CRED_IP_ADDRESS	[0]: 0x03050309
9	AZN_CRED_MECH_ID	[0]: IV_URAF_V3.0
10	AZN_CRED_PRINCIPAL_DOMAIN	[0]: Default
11	AZN_CRED_PRINCIPAL_NAME	[0]: emp1
12	AZN_CRED_PRINCIPAL_UUID	[0]: 9d5f1ea8-df36-11d9-872b-000c29a951ea
13	AZN_CRED_QOP_INFO	[0]: None
14	AZN_CRED_REGISTRY_ID	[0]: CN=Employee One,CN=Users,DC=bigcorp,DC=com
15	AZN_CRED_USER_INFO	[0]:
16	AZN_CRED_VERSION	[0]: 0x00000510
17	tagvalue_active_dir_cn	[0]: Employee One
18	tagvalue_active_dir_mail	[0]: emp1@bigcorp.com

Figure B-1 Sample epac from BigCorp

Testing XSL rules

There are a couple of very useful techniques for testing your XSL mapping rules prior to deploying them to Tivoli Federated Identity Manager.

First either write or acquire from an Tivoli Federated Identity Manager DEBUG_MAX trace log an example XML file of an STSUniversalUser that your XSL will operate on. The examples shown for each token type in “Mapping

between STSUniversalUser and native tokens” on page 384 should be a good starting point.

There is a command-line XSLT program available to run your XSL over the XML file. Starting with the XML of a sample Access Manager credential STSUU from B-2, and the mapping rule from use case 1 located in B-8, the command line execution is depicted at Example B-7. While the output is not all that pretty since the XML is not formatted, you can easily save it to a text file and view it in a better XML viewer. If you include your Java code in the classpath when executing the command-line tool, you can even test call Java from XSL on the command line. For more information on running XSLT from the command line, see:

<http://xml.apache.org/xalan-j/commandline.html>

Example: B-7 Testing XSLT from a command line

```
C:\temp>\Programs\1\websphere\appserver\java\bin\java
org.apache.xalan.xslt.Process -in emp1.xml -xsl bigcorp_mapping_1.xsl

<?xml version="1.0" encoding="utf-8"?>
<stsuser:STSUniversalUser xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
<stsuser:Principal>
<stsuser:Attribute type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress"
name="name">
<stsuser:Value>emp1@bigcorp.com</stsuser:Value>
</stsuser:Attribute>
</stsuser:Principal>
<GroupList xmlns="urn:ibm:names:ITFIM:1.0:stsuser">
<stsuser:Group name="employees" type="urn:ibm:names:ITFIM:5.1:accessmanager">
<stsuser:Attribute name="uuid" type="urn:ibm:names:ITFIM:5.1:accessmanager">
<stsuser:Value>6f0f791c-ea49-11d9-a4eb-000c29d2099e</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute name="registryid"
type="urn:ibm:names:ITFIM:5.1:accessmanager">
<stsuser:Value>CN=employees,CN=Users,DC=bigcorp,DC=com</stsuser:Value>
</stsuser:Attribute>
</stsuser:Group>
</GroupList>
<stsuser:AttributeList>
<stsuser:Attribute type="urn:oasis:names:tc:SAML:1.0:assertion"
name="AuthenticationMethod">
<stsuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuser:Value>
</stsuser:Attribute>
<stsuser:Attribute type="http://www.bigcorp.com/cn" name="cn">
<stsuser:Value>Employee One</stsuser:Value>
</stsuser:Attribute>
</stsuser:AttributeList>
```


Even more sophisticated is the Eclipse programming platform plug-ins for developing and executing XSLT rules. These plug-ins also appear in WebSphere Studio Application Developer, presently called Rational Software Developer and Rational Software Architect. This will allow you to step through your XSL command-by-command, and is an excellent way to debug XSL logic. Figure B-2 shows a screen from WebSphere Studio Application Developer while stepping through an XSL command for the same example shown on the command line.

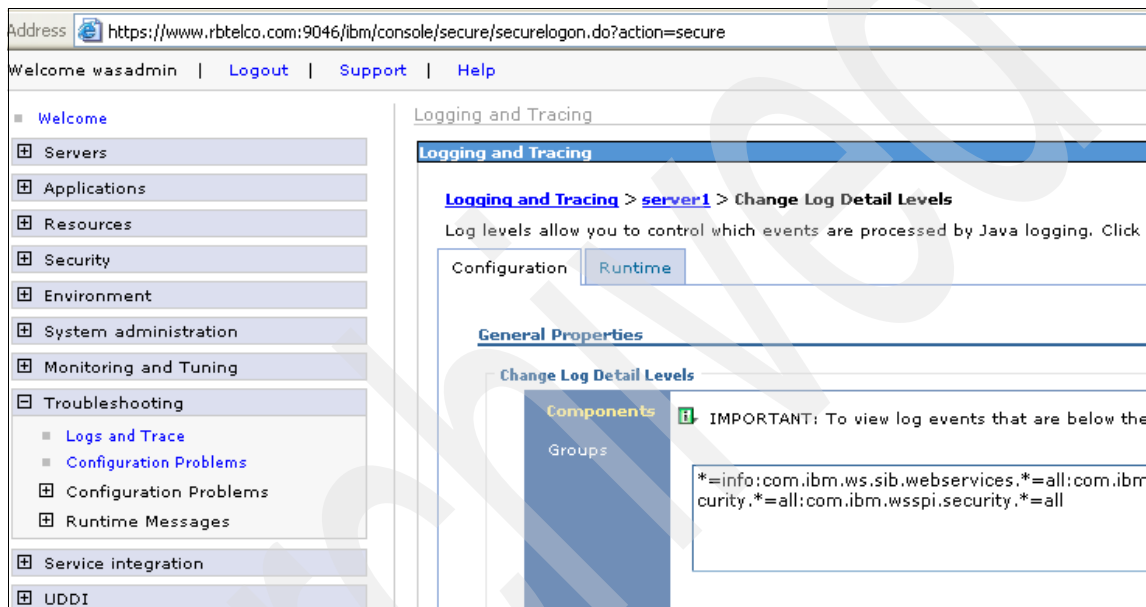


Figure B-2 Debugging XSLT with WebSphere Studio Application Developer (Eclipse Platform)

Scenario mapping rules

This section contains all the mapping rules and Java code used for the scenarios in this book.

Use case 1 mapping rules

This section contains the mapping rules used at BigCorp and RBTravel for Chapter 7, “Use case 1 - SAML/JITP” on page 193.

BigCorp mapping for use case 1

Example B-8 shows the mapping rule at BigCorp used to transform an Access Manager Credential into a SAML 1.0 Assertion.

Example: B-8 BigCorp mapping for use case 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser" version="1.0">

  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />

  <!--
    Initially we start with a copy of the document.
  -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!--
    This template replaces the entire Principal element with one that
    contains
    just the email address (from the ivcred tagvalue_activedir_mail) and the
    data type
    appropriate for SAML.
  -->
  <xsl:template match="//stsuser:Principal">
    <stsuser:Principal>
      <stsuser:Attribute name="name"
        type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">
        <stsuser:Value>
          <xsl:value-of
            select="//stsuser:AttributeList/stsuser:Attribute[@name='tagvalue_activedir_m
            ail'][@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuser:Value" />
          </stsuser:Value>
        </stsuser:Attribute>
      </stsuser:Principal>
    </xsl:template>

  <!--
    This template builds a new AttributeList. This involves:
    a) Adding an AuthenticationMethod attribute to meet SAML requirements.
    We assume
```

```

        this is always the "password" mechanism, regardless of what the TAM
credential
actually says.
    b) Map the tagvalue_cn to commonName
-->
<xsl:template match="//stsuser:AttributeList">
    <stsuser:AttributeList>

        <!-- First the authentication method attribute -->
        <stsuser:Attribute name="AuthenticationMethod"
            type="urn:oasis:names:tc:SAML:1.0:assertion">
            <stsuser:Value>
                urn:oasis:names:tc:SAML:1.0:am:password
            </stsuser:Value>
        </stsuser:Attribute>

        <!-- Now the cn attribute -->
        <stsuser:Attribute name="cn"
            type="http://www.bigcorp.com/cn">
            <stsuser:Value>
                <xsl:value-of

select="//stsuser:AttributeList/stsuser:Attribute[@name='tagvalue_activedir_c
n'][@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuser:Value" />
            </stsuser:Value>
        </stsuser:Attribute>

    </stsuser:AttributeList>
</xsl:template>

</xsl:stylesheet>

```

RBTravel mapping for use case 1

Example B-9 shows the mapping rule at RBTravel used to transform a SAML 1.0 assertion into an Access Manager credential. Note that this rule also calls out to Java code to just-in-time provisioning the user if necessary. B-10 shows the Java code that implemented this provisioning.

Example: B-9 RBTravel mapping for use case 1

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0" xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser"
    xmlns:xalan="http://xml.apache.org/xalan"
    xmlns:fromjava="JITProvisioning">

    <xalan:component prefix="fromjava" functions="doJITP">

```

```

        <xalan:script lang="javaclass"
            src="xalan://com.tivoli.am.fim.redbook.JITProvisioning" />
    </xalan:component>

    <xsl:strip-space elements="*" />
    <xsl:output method="xml" version="1.0" encoding="utf-8"
        indent="yes" />

    <!--
        Initially we start with a copy of the document.
    -->
    <xsl:template match="@* | node()">
        <xsl:copy>
            <xsl:apply-templates select="@* | node()" />
        </xsl:copy>
    </xsl:template>

    <!--
        This template replaces the AttributeList with one containing only the
subset
of attributes we are interested in (whilst modifying their data type)
and
adds a tagvalue_mail attribute which is the current principal name value
(from the SAML assertion). When copying attributes that we are
interested
in we also map their names as follows:
cn->tagvalue_cn
We also include an AUTHENTICATION_LEVEL attribute, with value 1.
    -->
    <xsl:template match="//stsuser:AttributeList">
        <stsuser:AttributeList>

            <!-- The tagvalue_cn attribute -->
            <stsuser:Attribute name="tagvalue_cn"
                type="urn:ibm:names:ITFIM:5.1:accessmanager">
                <stsuser:Value>
                    <xsl:value-of

select="//stsuser:AttributeList/stsuser:Attribute[@name='cn'][@type='http://w
ww.bigcorp.com/cn']/stsuser:Value" />
                </stsuser:Value>
            </stsuser:Attribute>

            <!-- The tagvalue_mail attribute -->
            <stsuser:Attribute name="tagvalue_mail"
                type="urn:ibm:names:ITFIM:5.1:accessmanager">
                <stsuser:Value>
                    <xsl:value-of

```

```

select="//stsuuser:Principal/stsuuser:Attribute[@name='name'][@type='urn:oasis:
names:tc:SAML:1.0:assertion#emailAddress']/stsuuser:Value" />
  </stsuuser:Value>
</stsuuser:Attribute>

  <!-- The AUTHENTICATION_LEVEL attribute -->
  <stsuuser:Attribute name="AUTHENTICATION_LEVEL"
    type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuuser:Value>1</stsuuser:Value>
  </stsuuser:Attribute>

</stsuuser:AttributeList>
</xsl:template>

<!--
  This will just-in-time provision the user if necessary, and update the
  principal name to the correct type.
  We still use the email address for the username.
-->
<xsl:template
  match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">

  <xsl:variable name="username">
    <xsl:value-of

select="//stsuuser:Principal/stsuuser:Attribute[@name='name'][@type='urn:oasis:
names:tc:SAML:1.0:assertion#emailAddress']/stsuuser:Value" />
    </xsl:variable>

    <xsl:variable name="tamConfigURL">

file:///opt/IBM/WebSphere/AppServer/profiles/rbtravel/config/itfim/rbtravel/nod
es/fimNode03Cell/rbtravel/server1/amconfig.conf
    </xsl:variable>

    <stsuuser:Attribute name="name"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuuser:Value>
        <xsl:value-of
          select="fromjava:doJITP($username,$tamConfigURL)" />
        </stsuuser:Value>
      </stsuuser:Attribute>

    </xsl:template>
  </xsl:stylesheet>

```

Example: B-10 Java code for just-in-time provisioning

```
package com.tivoli.am.fim.redbook;

import com.tivoli.pd.jutil.PDContext;
import com.tivoli.pd.jadmin.PDUser;
import com.tivoli.pd.jutil.PDRgyUserName;
import com.tivoli.pd.jutil.PDMessages;
import com.tivoli.pd.jutil.PDException;

import com.tivoli.mts.PDPrincipal;

import java.net.URL;
import java.net.MalformedURLException;

/**
 * @author Shane Weeden
 *
 */
public class JITProvisioning {

    static PDContext m_context = null;

    static final String NEWUSER_PREFIX = "cn=";

    static final String NEWUSER_SUFFIX = ",o=rbtravel,c=us";

    static final String NEWUSER_DUMMY_PWD = "passw0rd_never_used";

    String _tamConfigURL = null;

    public String doJITP(String username, String tamConfigURL) {
        _tamConfigURL = tamConfigURL;

        // protect the PDJRTE context
        synchronized (this.getClass()) {
            try {
                System.out.println("Checking for existing TAM user: "
                    + username);

                if (userExists(username)) {
                    System.out.println("Tam user found: " + username);
                } else {
                    createUser(username);
                }
            } catch (Exception e) {
                System.out
                    .println("Unexpected exception in JITProvisioning.doJITP( "
                        + username + "): " + e.getMessage());
            }
        }
    }
}
```

```

    }
}
return username;
}

private boolean userExists(String username) throws NullPointerException,
    IllegalArgumentException, SecurityException, MalformedURLException {
    boolean result = false;

    // use azn api rather than pdadmin api here for performance /
    // scalability reasons
    try {
        PDPrincipal principal = new PDPrincipal(username, getConfigURL());

        // no exception here, so user must exist
        result = true;

    } catch (IllegalStateException ise) {
        // this is the "normal case" when the user does not exist
        System.out.println("Tam user not found: " + username + " Details: "
            + ise.getMessage());
        result = false;
    }
    return result;
}

private void createUser(String username) throws PDException,
    MalformedURLException {
    // use pdadmin api to create the user
    initContext();
    try {
        PDMessages msgs = new PDMessages();
        PDUUser.createUser(m_context, username, new PDRgyUserName(
            NEWUSER_PREFIX + username + NEWUSER_SUFFIX, username,
            username), null, (new String(NEWUSER_DUMMY_PWD))
            .toCharArray(), null, false, true, msgs);

        System.out.println("User created: " + username + " with messages: "
            + msgs.toString());
        msgs.clear();

        // don't forget to set the account-valid to yes
        PDUUser.setAccountValid(m_context, username, true, msgs);
        System.out.println("User account set to valid: " + username
            + " with messages: " + msgs.toString());
        msgs.clear();
    } catch (PDException pde) {
        System.out.println("PDException caught while adding user: "
            + pde.toString());
    }
}

```

```

    }
}

private void initContext() throws PDException, MalformedURLException {
    if (m_context != null) {
        return;
    }
    m_context = new PDContext(getConfigURL());
}

private URL getConfigURL() throws MalformedURLException {
    return new URL(_tamConfigURL);
}
}
}

```

Use case 2 mapping rules

This section contains the mapping rules used at BigCorp and RBTelco for use case 2.

BigCorp mapping for use case 2

Example B-11 shows the mapping rule at BigCorp used to transform an Access Manager credential into a SAML 1.1 assertion used for WS-Federation login.

Example: B-11 BigCorp mapping for use case 2

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser" version="1.0">

  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />

  <!--
    Initially we start with a copy of the document.
  -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!--

```


This template replaces the entire Principal element with one that contains just the email address (from the ivcred tagvalue_activatedir_mail) and the data type appropriate for SAML.

```
-->
<xsl:template match="//stsuser:Principal">
  <stsuser:Principal>
    <stsuser:Attribute name="name"
      type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
      <stsuser:Value>
        <xsl:value-of
select="//stsuser:AttributeList/stsuser:Attribute[@name='tagvalue_activatedir_mail'][@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuser:Value" />
        </stsuser:Value>
      </stsuser:Attribute>
    </stsuser:Principal>
  </xsl:template>

<!--
  This template builds a new AttributeList. This involves:
  a) Adding an AuthenticationMethod attribute to meet SAML requirements.
We assume
  this is always the "password" mechanism, regardless of what the TAM
credential
  actually says.
  b) Map the tagvalue_cn to cn
-->
<xsl:template match="//stsuser:AttributeList">
  <stsuser:AttributeList>

    <!-- First the authentication method attribute -->
    <stsuser:Attribute name="AuthenticationMethod"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        urn:oasis:names:tc:SAML:1.0:am:password
      </stsuser:Value>
    </stsuser:Attribute>

    <!-- Now the cn attribute -->
    <stsuser:Attribute name="cn"
      type="http://www.bigcorp.com/cn">
      <stsuser:Value>
        <xsl:value-of
select="//stsuser:AttributeList/stsuser:Attribute[@name='tagvalue_activatedir_cn'][@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuser:Value" />
        </stsuser:Value>
      </stsuser:Attribute>
  </stsuser:AttributeList>
</xsl:template>
```

```

        </stsuser:Attribute>
    </stsuser:AttributeList>
</xsl:template>

</xsl:stylesheet>

```

RBTelco mapping for use case 2

Example B-12 shows the mapping rule at RBTelco used to transform a SAML 1.1 assertion into an Access Manager credential.

Example: B-12 RBTelco mapping for use case 2

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0" xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">

  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />

  <!--
    Initially we start with a copy of the document.
  -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!--
    This template replaces the AttributeList with one containing only the
    subset
    of attributes we are interested in (whilst modifying their data type)
    and
    adds a tagvalue_mail attribute which is the current principal name value
    (from the SAML assertion). When copying attributes that we are
    interested
    in we also map their names as follows:
    cn->tagvalue_cn

    We also add a static attribute which records the fact that this user is
    from bigcorp.
    This is used later in the stock quote application to determine that the
    user should
    get a realtime stock quote.

    We also include an AUTHENTICATION_LEVEL attribute, with value 1.
  -->

```

```

-->
<xsl:template match="//stsuser:AttributeList">
  <stsuser:AttributeList>

    <!-- The tagvalue_name attribute -->
    <stsuser:Attribute name="tagvalue_cn"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>
        <xsl:value-of

select="//stsuser:AttributeList/stsuser:Attribute[@name='cn'][@type='http://w
ww.bigcorp.com/cn']/stsuser:Value" />
        </stsuser:Value>
      </stsuser:Attribute>

    <!-- The tagvalue_mail attribute -->
    <stsuser:Attribute name="tagvalue_mail"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>
        <xsl:value-of

select="//stsuser:Principal/stsuser:Attribute[@name='name'][@type='urn:oasis:
names:tc:SAML:1.1:nameid-format:emailAddress']/stsuser:Value" />
        </stsuser:Value>
      </stsuser:Attribute>

    <!-- The tagvalue_fim_partner attribute -->
    <stsuser:Attribute name="tagvalue_fim_partner"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>BigCorp</stsuser:Value>
    </stsuser:Attribute>

    <!-- The AUTHENTICATION_LEVEL attribute -->
    <stsuser:Attribute name="AUTHENTICATION_LEVEL"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>1</stsuser:Value>
    </stsuser:Attribute>

  </stsuser:AttributeList>
</xsl:template>

<!--
This will replace the principal name (which was the email address in
the SAML assertion) with the user "me_mary".
-->
<xsl:template
  match="//stsuser:Principal/stsuser:Attribute[@name='name']">
  <stsuser:Attribute name="name"
    type="urn:ibm:names:ITFIM:5.1:accessmanager">

```

```

        <stsuser:Value>bigcorp_guest</stsuser:Value>
    </stsuser:Attribute>
</xsl:template>
</xsl:stylesheet>

```

Use case 3 mapping rules

This section contains the mapping rules used at RBTelco, RBBanking, and RBTickets for use case 3.

RBTelco mapping for use case 3

Example B-13 shows the mapping rule at RBTelco used to transform an Access Manager credential into a Liberty 1.2 used for sign-on to either RBBanking or RBTickets.

Example: B-13 RBTelco mapping for use case 3

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser" version="1.0">
  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />

  <!-- Initially we start with a copy of the document. -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!--
  This template updates the name type in the Principal element
  with a type appropriate for Liberty 1.2 Assertion.
  -->
  <xsl:template match="//stsuser:Principal">
    <stsuser:Principal>
      <stsuser:Attribute name="name"
        type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        <stsuser:Value>
          <xsl:value-of select="stsuser:Value" />
        </stsuser:Value>
      </stsuser:Attribute>
    </stsuser:Principal>
  </xsl:template>

```

```

<!--
  This template builds a new AttributeList.
  This involves adding an AuthenticationMethod attribute to meet SAML
  requirements.
  We assume this is always the "password" mechanism, regardless of what
  the TAM credential actually says.

  Note that in this rule, we don't send any extended attributes to the
  partner. This is typical of
  real-world liberty partnerships. Some commercial products cannot handle
  extended attributes in
  the SAML assertion.
-->
<xsl:template match="//stsuser:AttributeList">
  <stsuser:AttributeList>

    <!-- The authentication method attribute -->
    <stsuser:Attribute name="AuthenticationMethod"
      type="urn:oasis:names:tc:SAML:1.0:assertion">
      <stsuser:Value>
        urn:oasis:names:tc:SAML:1.0:am:password
      </stsuser:Value>
    </stsuser:Attribute>
  </stsuser:AttributeList>
</xsl:template>
</xsl:stylesheet>

```

RBBanking mapping for use case 3

Example B-14 shows the mapping rule at RBBanking used to transform a Liberty 1.2 assertion into an Access Manager credential.

Example: B-14 RBBanking mapping for use case 3

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0" xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">
  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />
  <!-- Initially we start with a copy of the document. -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!-- This will update the principal name type for TAM. -->

```

```

<xsl:template
  match="//stsuser:Principal/stsuser:Attribute[@name='name']">
  <stsuser:Attribute name="name"
    type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>
      <xsl:value-of select="stsuser:Value" />
    </stsuser:Value>
  </stsuser:Attribute>
</xsl:template>

<!--
  This template replaces the AttributeList with one containing an
  identifier to let us know
  this was a liberty login. This is not really needed, but shows adding
  extended attributes
  to the TAM credential.

  We also include an AUTHENTICATION_LEVEL attribute, with value 1.
-->
<xsl:template match="//stsuser:AttributeList">
  <stsuser:AttributeList>

    <!-- The tagvalue_fim_login attribute -->
    <stsuser:Attribute name="tagvalue_fim_login"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>Liberty12</stsuser:Value>
    </stsuser:Attribute>

    <!-- The AUTHENTICATION_LEVEL attribute -->
    <stsuser:Attribute name="AUTHENTICATION_LEVEL"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>1</stsuser:Value>
    </stsuser:Attribute>

  </stsuser:AttributeList>
</xsl:template>

</xsl:stylesheet>

```

RBTickets mapping for use case 3

Example B-15 shows the mapping rule at RBTickets used to transform a Liberty 1.2 assertion into an Access Manager credential.

Example: B-15 RBTickets mapping for use case 3

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0" xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser">

```

```

<xsl:strip-space elements="*" />
<xsl:output method="xml" version="1.0" encoding="utf-8"
  indent="yes" />
<!-- Initially we start with a copy of the document. -->
<xsl:template match="@* | node()">
  <xsl:copy>
    <xsl:apply-templates select="@* | node()" />
  </xsl:copy>
</xsl:template>

<!-- This will update the principal name type for TAM. -->
<xsl:template
  match="//stsuser:Principal/stsuser:Attribute[@name='name']">
  <stsuser:Attribute name="name"
    type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuser:Value>
      <xsl:value-of select="stsuser:Value" />
    </stsuser:Value>
  </stsuser:Attribute>
</xsl:template>

<!--
  This template replaces the AttributeList with one containing an
  identifier to let us know
  this was a liberty login. This is not really needed, but shows adding
  extended attributes
  to the TAM credential.

  We also include an AUTHENTICATION_LEVEL attribute, with value 1.
-->
<xsl:template match="//stsuser:AttributeList">
  <stsuser:AttributeList>

    <!-- The tagvalue_fim_login attribute -->
    <stsuser:Attribute name="tagvalue_fim_login"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>Liberty12</stsuser:Value>
    </stsuser:Attribute>

    <!-- The AUTHENTICATION_LEVEL attribute -->
    <stsuser:Attribute name="AUTHENTICATION_LEVEL"
      type="urn:ibm:names:ITFIM:5.1:accessmanager">
      <stsuser:Value>1</stsuser:Value>
    </stsuser:Attribute>

  </stsuser:AttributeList>
</xsl:template>

```

Use case 4 mapping rules

This section contains the mapping rules used at RBTelco and RBStocks for use case 4, described in Chapter 10, “Use case 4 - Web services security management” on page 291.

RBTelco mapping for use case 4

Example B-16 shows the mapping rule at RBTelco used to transform an Access Manager credential into a SAML 1.0 assertion used for the WS-Security token representing the caller of the stock quote Web service.

Example: B-16 RBTelco mapping for use case 4

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser" version="1.0">

  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />

  <!--
    Initially we start with a copy of the document.
  -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!--
    This template replaces the entire Principal element with one that
    contains
    the email address of the user.
  -->
  <xsl:template match="//stsuuser:Principal">
    <stsuuser:Principal>
      <stsuuser:Attribute name="name"
        type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        <stsuuser:Value>
          <xsl:value-of
```



```

select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_mail'][@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
  </stsuuser:Attribute>
</stsuuser:Principal>
</xsl:template>

<!--
  This template builds a new AttributeList. This involves:
  a) Adding an AuthenticationMethod attribute to meet SAML requirements.
We assume
  this is always the "password" mechanism, regardless of what the TAM
credential
  actually says.
  b) Create a user_home attribute which indicates where this user's home
registry is.
  The way we determine the user's home registry is from the TAM attribute
tagvalue_fim_partner.
  If this attribute exists in the TAM credential, it's value represents
the company the user has
  performed a federated SSO from. If it doesn't exist, we assume the user
logged in locally to
  rbtelco, and set the value to that.
-->
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>

    <!-- First the authentication method attribute -->
    <stsuuser:Attribute name="AuthenticationMethod"
      type="urn:oasis:names:tc:SAML:1.0:assertion">

<stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
    </stsuuser:Attribute>

    <!-- Now the user_home attribute -->
    <stsuuser:Attribute name="user_home"
      type="http://rbtelco.com/user_home">
    <xsl:choose>
      <xsl:when
test="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_fim_partner']
[@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value">
      <stsuuser:Value>
        <xsl:value-of

select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_fim_partner']
[@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>

```

```

        </xsl:when>
        <xsl:otherwise>
            <stsuser:Value>RBTelco</stsuser:Value>
        </xsl:otherwise>
    </xsl:choose>
</stsuser:Attribute>

    </stsuser:AttributeList>
</xsl:template>

</xsl:stylesheet>

```

RBStocks mapping for use case 4

Example B-17 shows the mapping rule at RBStocks used to transform a SAML 1.0 assertion into another SAML 1.0 assertion. Note that this rule performs a many-to-few user identity mapping, and also calls out to Java code to check the e-mail address of the user against a simple flat-file blacklist. B-18 shows the Java code that implemented this blacklist checking.

Example: B-17 RBStocks mapping for use case 4

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser"
  xmlns:xalan="http://xml.apache.org/xalan"
  xmlns:blacklist="http://www.rbstocks.com/blacklist" version="1.0">

  <xalan:component prefix="blacklist" functions="isBlacklisted">
    <xalan:script lang="java" class="com.tivoli.am.fim.redbook.Blacklist" />
  </xalan:component>

  <xsl:strip-space elements="*" />
  <xsl:output method="xml" version="1.0" encoding="utf-8"
    indent="yes" />

  <!--
    Initially we start with a copy of the document.
  -->
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>

  <!--

```

This template replaces the entire Principal element with one that contains either "realtime", "delayed" or "blacklisted" according to the following psuedo-code:

```

if ( isBlacklisted( email, filename ) )
{
  username = "blacklisted";
}
else if ( user_home == "RBTelco" )
{
  username = "delayed";
}
else
{
  username = "realtime";
}
-->
<xsl:template match="//stsuser:Principal">

  <xsl:variable name="email">
    <xsl:value-of

select="//stsuser:Principal/stsuser:Attribute[@name='name'][@type='urn:oasis:
names:tc:SAML:1.1:nameid-format:emailAddress']/stsuser:Value" />
    </xsl:variable>

  <xsl:variable name="user_home">
    <xsl:value-of

select="//stsuser:AttributeList/stsuser:Attribute[@name='user_home'][@type='h
ttp://www.rbtelco.com/user_home']/stsuser:Value" />
    </xsl:variable>

  <xsl:variable name="blacklistfile">
    /opt/IBM/FIM/apps/wsm/stockquote/blacklist.txt
  </xsl:variable>

  <xsl:variable name="isBlacklisted">
    <xsl:value-of
      select="blacklist:isBlacklisted($email,$blacklistfile)" />
    </xsl:variable>

  <stsuser:Principal>
    <stsuser:Attribute name="name"
      type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
      <stsuser:Value>
        <xsl:choose>
          <xsl:when test="$isBlacklisted = 'true'">
            blacklisted

```

```

        </xsl:when>
        <xsl:when test="$user_home = 'RBTelco'">
            delayed
        </xsl:when>
        <xsl:otherwise>realtime</xsl:otherwise>
    </xsl:choose>
    </stsuser:Value>
</stsuser:Attribute>
</stsuser:Principal>
</xsl:template>

<!--
    This template builds a new AttributeList containing just an
    AuthenticationMethod
    attribute to meet SAML requirements.
-->
<xsl:template match="//stsuser:AttributeList">
    <stsuser:AttributeList>

        <!-- The authentication method attribute -->
        <stsuser:Attribute name="AuthenticationMethod"
            type="urn:oasis:names:tc:SAML:1.0:assertion">

<stsuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuser:Value>
        </stsuser:Attribute>
    </stsuser:AttributeList>
</xsl:template>

</xsl:stylesheet>

```

Example: B-18 Java code for blacklist checking at RBStocks

```

package com.tivoli.am.fim.redbook;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.HashSet;

/**
 * @author Shane Weeden
 */
public class Blacklist {
    public String isBlacklisted(String email, String fileName) {
        Boolean result = null;
        HashSet blacklist = new HashSet();

        System.out.println("Checking blacklist file: " + fileName
            + " for email: " + email);
    }
}

```

```

if (email == null || fileName == null) {
    return Boolean.TRUE.toString();
} else {
    try {
        FileInputStream fis = new FileInputStream(fileName);
        InputStreamReader isr = new InputStreamReader(fis);
        BufferedReader br = new BufferedReader(isr);

        String line = br.readLine();
        while (line != null) {
            blacklist.add(line);
            line = br.readLine();
        }
        br.close();
        isr.close();
        fis.close();

        result = Boolean.valueOf(blacklist.contains(email));
    } catch (Exception e) {
        System.out.println("Exception caught checking blacklist: "
            + e.getMessage());

        // err on the safe side and blacklist this user
        result = Boolean.TRUE;
    }
}

System.out.println("Email: " + email + " Blacklist: " + fileName
    + " Result: " + result.toString());
return result.toString();
}
}

```

Archived

Keys and certificates

This appendix describes the keys and certificates that were generated for the use cases described in this book. Screen images of importing keys into Tivoli Federated Identity Manager are included for one of the companies in the use cases.

Keys and certificates

When designing the use cases for this book, one of the primary considerations was the various public/private key pairs that would be used to establish trust between the partners, and the keyfiles needed by each company. This section describes the key strategy for the book scenarios.

Required keys

Having established a certificate authority for the lab, keys were generated according to the various signing and encryption requirements for the use cases. These keys and their uses are summarized in Table C-1.

Table C-1 Keys for book use cases

Company name	Key alias	Key name	Uses
ALL	redbook_ca	cn=fim.redbook.ibm.com,o=ibm,c=us	Overall signing CA for all other certificates.
BigCorp	bigcorp_www	cn=www.bigcorp.com,o=bigcorp,c=us	Web Server server certificate - installed on both www and soap instances of WebSEAL.
	bigcorp_rbtravel	cn=bigcorp_rbtravel.bigcorp.com,o=bigcorp,c=us	Used by BigCorp for signing SAML 1.0 assertions and sample responses to RBTravel.
	bigcorp_rbtelco	cn=bigcorp_rbtravel.bigcorp.com,o=bigcorp,c=us	Used by BigCorp for signing SAML 1.1 assertion for ws-federation to RBTelco.

Company name	Key alias	Key name	Uses
RBTravel	rbtravel_www	cn=www.rbtravel.com,o=rbtravel,c=us	Web Server server certificate for www.rbtravel.com.
	rbtravel_bigcorp	cn=rbtravel_bigcorp.rbtravel.com,o=rbtravel,c=us	Used by RBTravel as a client certificate to connect to SAML/SOAP WebSEAL on BigCorp.
RBTelco	rbtelco_www	cn=www.rbtelco.com,o=rbtelco,c=us	Web Server server certificate for www.rbtravel.com.
	rbtelco_liberty	cn=rbtelco_liberty.rbtelco.com,o=rbtelco,c=us	Used by RBTelco for signing liberty assertions and messages to RBBanking and RBTickets. Note that for a single liberty federation only one signing key can be specified, not a separate key for each partner.
	rbtelco_rbstocks	cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us	Used by RBTelco for signing ws-security messages (and the saml assertion contained within it) to RBStocks. Note that for doing XML Encryption we will use RBStocks' public key.

Company name	Key alias	Key name	Uses
RBBanking	rbbanking_www	cn=www.rbbanking.com,o=rbbanking,c=us	Web Server server certificate for www.rbbanking.com.
	rbbanking_rbtelco	cn=rbbanking_rbtelco.rbbanking.com,o=rbbanking,c=us	Used by RBBanking for signing liberty messages to RBTelco.
RBTickets	rbtickets_www	cn=www.rbtickets.com,o=rbtickets,c=us	Web Server server certificate for www.rbtickets.com.
	rbtickets_rbtelco	cn=rbtickets_rbtelco.rbtickets.com,o=rbtickets,c=us	Used by RBTickets for signing liberty messages to RBTelco.
RBStocks	rbstocks_rbtelco	cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us	Used by RBStocks for signing ws-security response messages to RBTelco.

Keystore layout

The next job was to gather these keys into keystores that would be useful for the various companies in the scenario. Developing a pattern for storing the keys provides a consistent, predictable way to determine which keys should go in which keystores.

For the scenarios in this book, each company will potentially have five keystore files that can contain the above keys. Not all companies need every type of keystore.

The first is the pdsrv.kdb used by the WebSEAL server. This contains the Web server certificate and private key, and for those Web servers receiving authentication from SSL clients with client certificates, it will also contain the certificate authority certificate for the signer of those client certificates.

The other four keystores follow this pattern, and are all used by the Federated Identity Manager software:

- ▶ *companyname*-signing.jks contains private keys used for signing objects sent to partners. This includes signing assertions, SAML, or Liberty requests/responses, and Web services requests and responses.
- ▶ *companyname*-partners.jks contains public certificates of partner signing keys for signed objects received from partners. These are used to verify signatures on things like Liberty/SAML assertions and Liberty/SAML requests and responses and Web services requests and responses. They are also used to do XML Encryption for WS-security messages. That is, with XML Encryption, you encrypt a message for your partner using their public key.
- ▶ *companyname*-ca.jks contains the CA certificates of Web servers of partners. This is primarily used by the SOAP client portions of the FIM configuration to validate that they are talking to the correct server when sending SOAP requests.
- ▶ *companyname*-clients.jks contains the SSL client certificates used by those SOAP client portions of the FIM configuration that need to communicate via mutually authenticated SSL to a partner.

Keystores for BigCorp

Table C-2 shows the keystore files needed for BigCorp.

Table C-2 Keystores for BigCorp

Keystore name	Keys and certificates (alias)	Public certificate or private key
pdsrv.kdb (Used for both WebSEAL's which run on BigCorp)	bigcorp_www	Private
	redbook_ca	Public
bigcorp-signing.jks	bigcorp_rbtravel	Private
	bigcorp_rbtelco	Private
bigcorp-partners.jks	NOT NEEDED	
bigcorp-ca.jks	NOT NEEDED	
bigcorp-clients.jks	NOT NEEDED	

Keystores for RBTravel

Table C-3 on page 430 shows the keystore files needed for RBTravel.

Table C-3 Keystores for RBTravel

Keystore name	Keys and certificates (alias)	Public certificate or private key
pdsrv.kdb	rbtravel_www	Private
	redbook_ca	Public
rbtravel-signing.jks	NOT NEEDED	
rbtravel-partners.jks	NOT NEEDED	
rbtravel-ca.jks	redbook_ca	Public
rbtravel-clients.jks	rbtravel_bigcorp	Private

Keystores for RBTelco

Table C-4 shows the keystore files needed for RBTelco.

Table C-4 Keystores for RBTelco

Keystore name	Keys and certificates (alias)	Public certificate or private key
pdsrv.kdb	rbtelco_www	Private
	redbook_ca	Public
rbtelco-signing.jks	rbtelco_liberty	Private
	rbtelco_rbstocks	Private
rbtelco-partners.jks	bigcorp_rbtelco	Public
	rbbanking_rbtelco	Public
	rbtickets_rbtelco	Public
	rbstocks_rbtelco	Public
rbtelco-ca.jks	redbook_ca	
rbtelco-clients.jks	NOT NEEDED	

Keystores for RBBanking

Table C-5 on page 431 shows the keystore files needed for RBBanking.

Table C-5 Keystores for RBBanking

Keystore name	Keys and certificates (alias)	Public certificate or private key
pdsrv.kdb	rbbanking_www	Private
	redbook_ca	Public
rbbanking-signing.jks	rbbanking_rbtelco	Private
rbbanking-partners.jks	rbtelco_liberty	Public
rbbanking-ca.jks	redbook_ca	Public
rbbanking-clients.jks	NOT NEEDED	

Keystores for RBTickets

Table C-6 shows the keystore files needed for RBTickets.

Table C-6 Keystores for RBTickets

Keystore name	Keys and certificates (alias)	Public certificate or private Key
pdsrv.kdb	rbtickets_www	Private
	redbook_ca	Public
rbtickets-signing.jks	rbtickets_rbtelco	Private
rbtickets-partners.jks	rbtelco_liberty	Public
rbtickets-ca.jks	redbook_ca	Public
rbtickets-clients.jks	NOT NEEDED	

Keystores for RBStocks

Table C-7 shows the keystore files needed for RBStocks.

Table C-7 Keystores for RBStocks

Keystore name	Keys and certificates (alias)	Public certificate or private key
pdsrv.kdb	NOT NEEDED	
rbstocks-signing.jks	rbstocks_rbtelco	Private
rbstocks-partners.jks	rbtelco_rbstocks	Public

Keystore name	Keys and certificates (alias)	Public certificate or private key
rbstocks-ca.jks	NOT NEEDED	
rbstocka-clients.jks	NOT NEEDED	

Importing keys

This section demonstrates the use of the Tivoli Federated Identity Manager Console for importing keys into Tivoli Federated Identity Manager. Since the pattern is very repetitive, we only show the importing of the BigCorp signing keys.

Before importing the keys into the Tivoli Federated Identity Manager runtime, create the bigcorp-signing.jks file mentioned above containing the bigcorp_rbtravel and bigcorp_rbtelco private keys.

Log in to the Tivoli Federated Identity Manager Console, and navigate to Tivoli Federated Identity Manager → Service Management → Key Service. Figure C-1 on page 433 shows the screen you will see, with the default keystores.

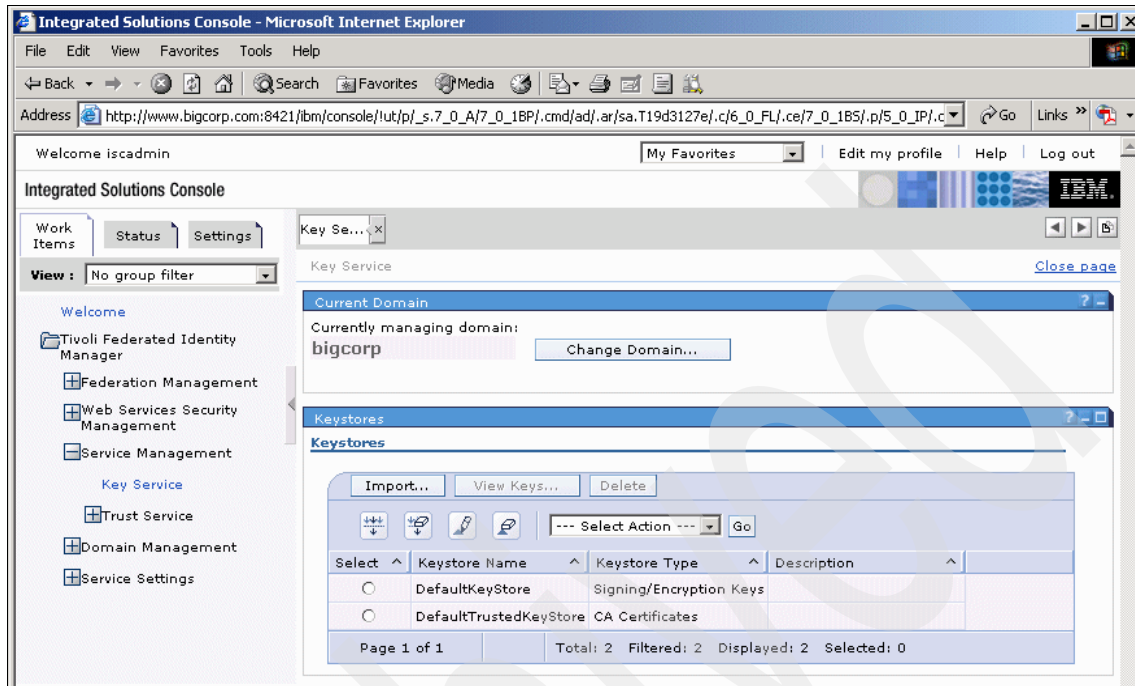


Figure C-1 Key Service Interface

Browse to the path of your bigcorp-signing.jks file, and enter your keystore password and the name of the file you want (without an extension) to save these keys into. Figure C-2 on page 434 shows what your screen should look like. Note that if you were importing a keystore that only contained public certificates (such as *companyname_partners* in our pattern), then you would select **CA Certificates** as the type. The difference with a CA Certificates keystore is that when selecting these certificates later during federation configurations, no keystore password is required.

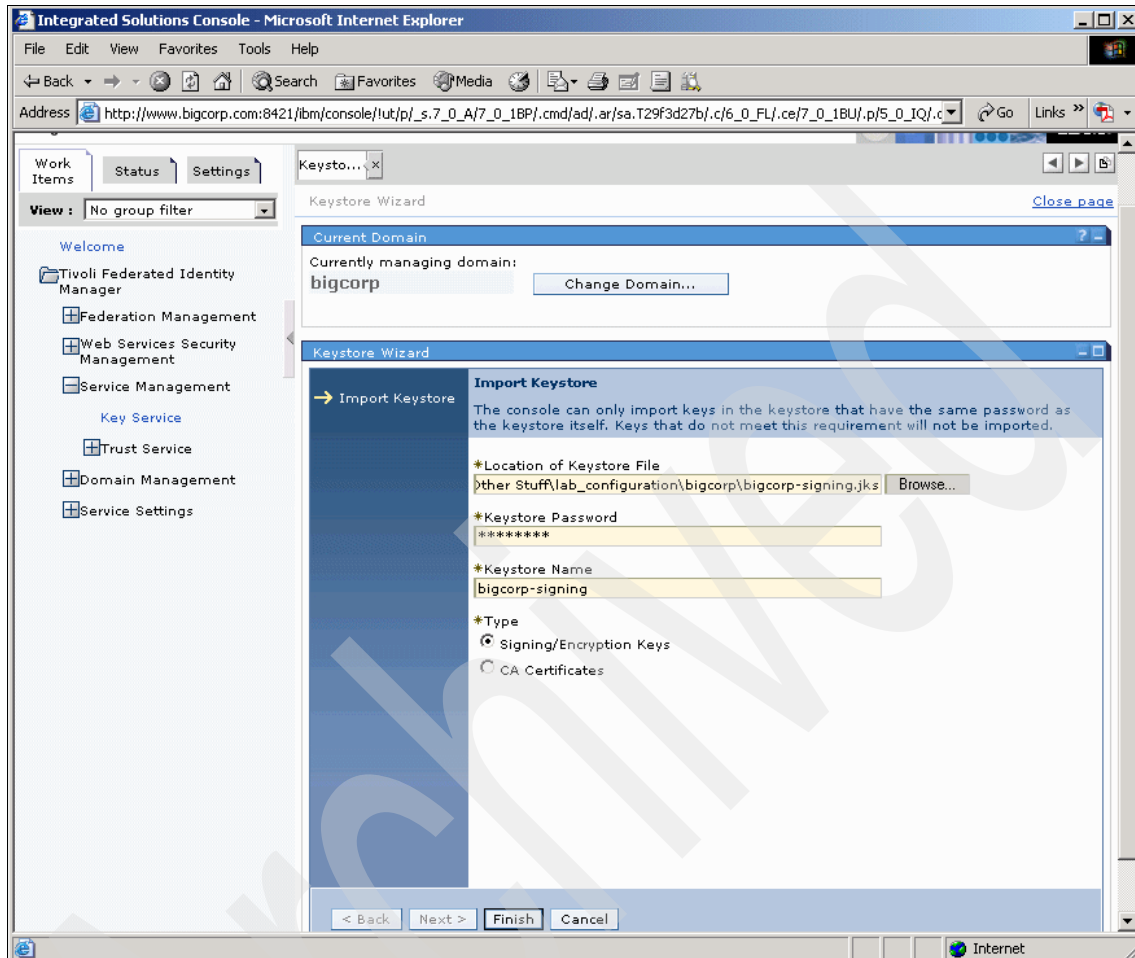


Figure C-2 Importing a keystore

Click **Finish** when complete, and the keys should be imported. Figure C-3 on page 435 shows the Key Service management screen again with your newly created key store, reminding you that a WebSphere restart is required to propagate changes to all nodes in the cluster.

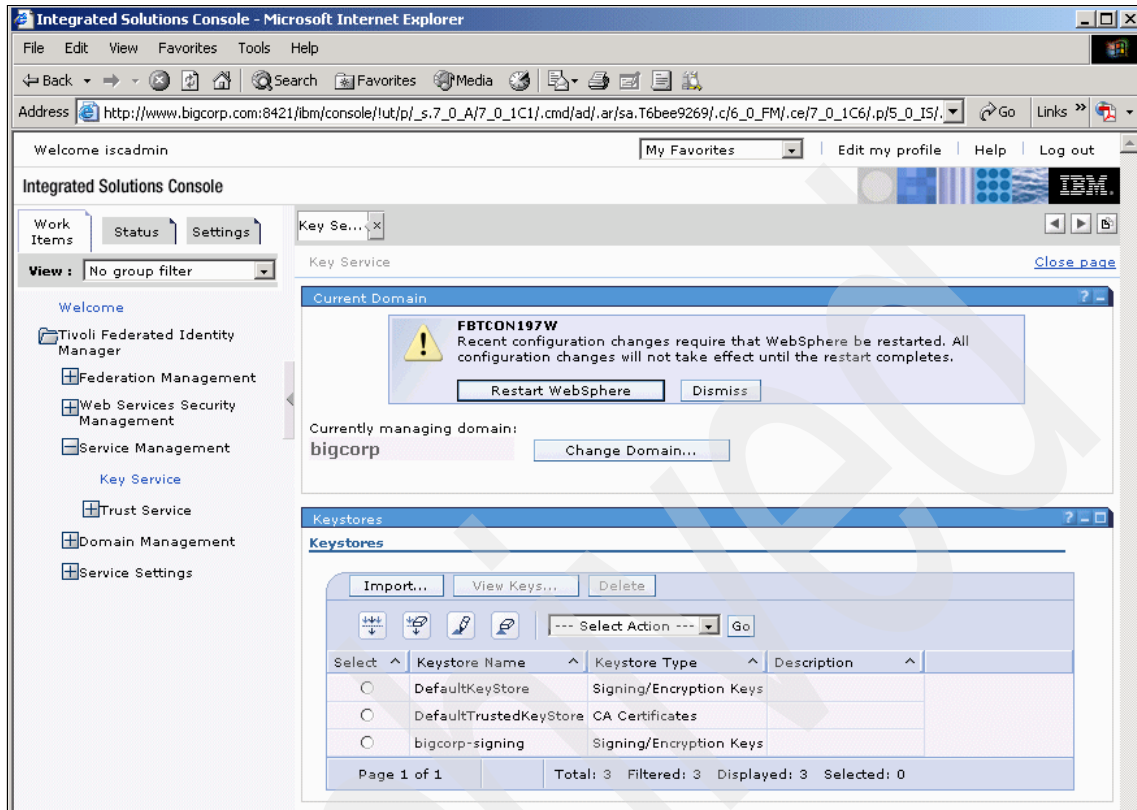


Figure C-3 Keystore import completed

Archived

WS-Security deployment descriptors

This appendix contains the WS-Security deployment descriptors used at the various WS-Security integration points in Chapter 10, “Use case 4 - Web services security management” on page 291.

IBM does not support the format of these deployment descriptors, or guarantee that they or their method of configuration via Rational Application Developer, WebSphere Application Server Toolkit, and the WebSphere Administration Console will remain the same from release to release; however, by perusing these deployment descriptors you can decipher precisely how the WS-Security configuration was achieved for the use case in this book’s development lab. This is very valuable information if you are trying to recreate this or a similar scenario in your own environment.

Web services client at RBTelco

The Web services client at RBTelco is invoked from a JAAS-protected JSP. WebSphere is configured with TAI++ for authentication via WebSEAL. More information on this configuration is in Chapter 10, “Use case 4 - Web services security management” on page 291.

This section shows the WS-Security extension and binding configuration for the jsp client. The requirement here is to use WS-Security with Tivoli Federated Identity Manager Web services security management components to retrieve the Access Manager credential from the current JAAS subject, and insert it as a security token in the request to the client-side Web services gateway.

RBTelco client extension configuration

Example D-1 shows the WS-Security client extension configuration for the stock quote Web service client invoked from RBTelco.

Example: D-1 RBTelco WS-Security client extension

```
<com.ibm.etools.webservice.wssect:WsClientExtension xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"

xmlns:com.ibm.etools.webservice.wssect="http://www.ibm.com/websphere/appserver/
schemas/5.0.2/wssect.xmi"
  xmi:id="WsClientExtension_1119298337875">
    <serviceRefs xmi:id="ServiceRef_1119451733515"
      serviceRefLink="service/StockQuoteServiceService">
      <portQnameBindings xmi:id="PortQnameBinding_1119451733515"
        portQnameNamespaceLink="http://StockQuote"
        portQnameLocalNameLink="StockQuoteService">
        <clientServiceConfig
          xmi:id="ClientServiceConfig_1119451733515">
          <securityRequestGeneratorServiceConfig
            xmi:id="SecurityRequestGeneratorServiceConfig_1119451733515">
            <securityToken xmi:id="SecurityToken_1119460896890"
              name="TAMToken"

uri="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"

              localName="BinarySecurityToken" />
            </securityRequestGeneratorServiceConfig>
          <securityResponseConsumerServiceConfig
            xmi:id="SecurityResponseConsumerServiceConfig_1119457591437" />
          </clientServiceConfig>
        </portQnameBindings>
```

```
</serviceRefs>  
</com.ibm.etools.webservice.wscontext:WsClientExtension>
```

RBTelco client binding configuration

Example D-2 shows the WS-Security client binding configuration for the stock quote Web service client invoked from RBTelco.

Example: D-2 RBTelco WS-Security client binding

```
<com.ibm.etools.webservice.wscbnd:ClientBinding xmi:version="2.0"  
xmlns:xmi="http://www.omg.org/XMI"  
xmlns:com.ibm.etools.webservice.wscbnd="http://www.ibm.com/websphere/appserver/  
schemas/5.0.2/wscbnd.xmi" xmi:id="ClientBinding_1119298337828">  
  <serviceRefs xmi:id="ServiceRef_1119451733484"  
serviceRefLink="service/StockQuoteServiceService">  
    <portQnameBindings xmi:id="PortQnameBinding_1119451733484"  
portQnameNamespaceLink="http://StockQuote"  
portQnameLocalNameLink="StockQuoteService">  
      <securityRequestGeneratorBindingConfig  
xmi:id="SecurityRequestGeneratorBindingConfig_1119451733484">  
        <tokenGenerator xmi:id="TokenGenerator_1120748320734"  
name="TAMTokenGenerator"  
classname="com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator">  
          <valueType xmi:id="ValueType_1120748320734"  
localName="BinarySecurityToken"  
uri="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-  
1.0.xsd" name=""/>  
          <callbackHandler xmi:id="CallbackHandler_1120748320734"  
classname="com.tivoli.am.fim.wssm.callbackhandlers.TAMTAICallbackHandler">  
            <properties xmi:id="Property_1120748320734"  
name="pdjrte.config.file" value="/opt/IBM/FIM/apps/wssm/wssm_pdjрте.config"/>  
              <basicAuth xmi:id="BasicAuth_1120748320734"/>  
            </callbackHandler>  
            <properties xmi:id="Property_1120748320735" name="trust.service.call"  
value="false"/>  
            <properties xmi:id="Property_1120748320736" name="default.issuer.uri"  
value="http://www.rbtelco.com/tamtai"/>  
            <partReference xmi:id="PartReference_1120748320734" part="TAMToken"/>  
          </tokenGenerator>  
        </securityRequestGeneratorBindingConfig>  
      </portQnameBindings>  
    </serviceRefs>  
</com.ibm.etools.webservice.wscbnd:ClientBinding>
```

Web services gateway at RBTelco

The Web services gateway (WSGW) at RBTelco receives the WS-Security message containing an Access Manager credential from the jsp-client. On the receiving (server) side, it performs a JAAS login using the Access Manager credential received in the security header. On the sending (client) side, it exchanges the Access Manager credential for a signed SAML assertion, then signs and encrypts the token and message body in the outbound message. More information on this configuration is contained in Chapter 10, "Use case 4 - Web services security management" on page 291.

This section shows the WS-Security configuration from the Web services gateway use to meet the server and client requirements described above. The Web services gateway uses a different format of deployment descriptor from standalone Web services client and server applications. The gateway has just one file that contains all of the client and server extension and binding configurations. Rather than just show the one large file here, we split it up and show those pieces within it that correspond to the logical server and client components of the gateway.

RBTelco WSGW server configuration

This section shows the WS-Security extension and binding components used for processing an inbound message at the RBTelco Web services gateway. The message originates from the jsp client and should contain an Access Manager credential as a security token in the header.

RBTelco WSGW server extension configuration

Example D-3 shows the WS-Security server extension configuration for the inbound side of the Web services gateway at RBTelco. Notice that both the required security token and caller part indicate an Access Manager credential.

Example: D-3 RBTelco Web services gateway inbound server extension

```
<sibwssecurity:SIBWSSecurityInboundConfig
  xmi:id="SIBWSSecurityInboundConfig_1120601734621"
  name="StockQuoteServiceInboundFinal">
  <serverServiceConfig xmi:id="ServerServiceConfig_1120601734621">
    <securityRequestConsumerServiceConfig
      xmi:id="SecurityRequestConsumerServiceConfig_1120601734621">
      <caller xmi:id="Caller_1120601896710" name="TAMCredential"
        localName="http://ibm.com/2004/01/itfim/ivcred" />
      <requiredSecurityToken
        xmi:id="RequiredSecurityToken_1120601850771" name="TAMCredential"
```

```

        localName="http://ibm.com/2004/01/itfim/ivcred" usage="Required"
    />
    </securityRequestConsumerServiceConfig>
    <securityResponseGeneratorServiceConfig
        xmi:id="SecurityResponseGeneratorServiceConfig_1120601734621" />
    </serverServiceConfig>
</sibwssecurity:SIBWSecurityInboundConfig>

```

RBTelco WSGW server binding configuration

Example D-4 shows the WS-Security server binding configuration for the inbound side of the Web services gateway at RBTelco.

Example: D-4 RBTelco Web services gateway inbound server binding

```

<securityRequestConsumerBindingConfig
xmi:id="SecurityRequestConsumerBindingConfig_1120761749718">
    <tokenConsumer xmi:id="TokenConsumer_1120761845347"
        classname="com.tivoli.am.fim.wssm.tokenconsumers.WSSMTokenConsumer"
        name="TAMCredential">
        <valueType xmi:id="ValueType_1120761845347"
            localName="http://ibm.com/2004/01/itfim/ivcred" uri="" name="" />
        <jAASConfig xmi:id="JAASConfig_1120762111306"
            configName="system.itfim.wssm.tamcredential">
            <properties xmi:id="Property_1120793343454" name="pdjrte.config.file"
                value="/opt/IBM/FIM/wsgw/wssm/wssm_pdjrtc.config"/>
            </jAASConfig>
            <properties xmi:id="Property_1120761889274" name="trust.service.call"
                value="false"/>
            <partReference xmi:id="PartReference_1120761845347"
                part="TAMCredential" name="" />
        </tokenConsumer>
    </securityRequestConsumerBindingConfig>
</sibwssecurity:SIBWSecurityRequestConsumerBindingConfig>

```

RBTelco WSGW client configuration

This section shows the WS-Security extension and binding components used for processing an outbound message at the RBTelco Web services gateway. The message is being prepared for sending to RBStocks, and must exchange the Access Manager credential for a signed SAML assertion, then use WS-Security signing and encryption to sign and encrypt the combination of the SAML assertion and message body. This “binds” the token to the message body. On the response from RBStocks, we require the message body to be signed and encrypted.

RBTelco WSGW client extension configuration

Example D-5 shows the WS-Security client extension configuration for the outbound side of the Web services gateway at RBTelco. Both the request and response requirements are shown, though the request is the piece of primary interest to us.

Example: D-5 RBTelco Web services gateway outbound client extension

```
<clientServiceConfig xmi:id="ClientServiceConfig_1119712855294">
  <securityRequestGeneratorServiceConfig
xmi:id="SecurityRequestGeneratorServiceConfig_1119712855294">
  <integrity xmi:id="Integrity_1119713061835" name="BODY" order="1">
    <messageParts xmi:id="MessageParts_1119713079023"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
name="body" keyword="body"/>
  </integrity>
  <integrity xmi:id="Integrity_1120069694880" name="TOKEN" order="2">
    <messageParts xmi:id="MessageParts_1120069771940"
Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116" name="SAML"
keyword=""/>
  </integrity>
  <confidentiality xmi:id="Confidentiality_1119714405086"
name="BODY_AND_TOKEN" order="3">
    <messageParts xmi:id="MessageParts_1119714456729"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
name="bodycontent" keyword="bodycontent"/>
    <messageParts xmi:id="MessageParts_1121304044138"
Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116" name="token"
keyword="/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and
local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/env
elope/' and
local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/0
1/oasis-200401-wss-wssecurity-secext-1.0.xsd' and
local-name()='Security']/*[namespace-uri()='urn:oasis:names:tc:SAML:1.0:asserti
on' and local-name()='Assertion']"/>
  </confidentiality>
  <securityToken xmi:id="SecurityToken_1119712891121" name="SAML"
uri="urn:oasis:names:tc:SAML:1.0:assertion" localName="Assertion"/>
</securityRequestGeneratorServiceConfig>
  <securityResponseConsumerServiceConfig
xmi:id="SecurityResponseConsumerServiceConfig_1119712855295">
  <requiredIntegrity xmi:id="RequiredIntegrity_1119716404637"
name="required_integrity" usage="Required">
    <messageParts xmi:id="MessageParts_1119716418295"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
name="body" keyword="body"/>
  </requiredIntegrity>
```



```

        <requiredConfidentiality xmi:id="RequiredConfidentiality_1119716473735"
name="required_confidentiality" usage="Required">
        <messageParts xmi:id="MessageParts_1119716487035"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
name="bodycontent" keyword="bodycontent"/>
    </requiredConfidentiality>
    </securityResponseConsumerServiceConfig>
</clientServiceConfig>

```

RBTelco WSGW client binding configuration

Example D-6 shows the WS-Security client binding configuration for the outbound side of the Web services gateway at RBTelco. Of particular interest in this section is the SigningInfo transform used on the SAML assertion. Use of the default transform causes a wsu:Id attribute to be inserted into the XML element being signed (in our case the SAML assertion). This actually invalidates the SAML assertion, since it now contains an attribute that is not part of the SAML schema. Since we needed a SigningInfo reference URI that did not utilize the wsu:Id, the transform depicted in this binding was used.

Example D-7 on page 446 shows the response binding; this shows the keys, and so on used when the response body is signed and encrypted.

Example: D-6 RBTelco Web services gateway client binding to RBStocks

```

<securityRequestGeneratorBindingConfig
xmi:id="SecurityRequestGeneratorBindingConfig_1119713176727"
wsseNameSpace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <signingInfo xmi:id="SigningInfo_1119714258006"
name="wssm_client_signinfo">
        <signatureMethod xmi:id="SignatureMethod_1119714258006"
algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <canonicalizationMethod xmi:id="CanonicalizationMethod_1119714258006"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <partReference xmi:id="PartReference_1119714293006" part="BODY"
name="BODY">
            <transform xmi:id="Transform_1119714311844"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" name="body_transform"/>
            <digestMethod xmi:id="DigestMethod_1119714293006"
algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        </partReference>
        <partReference xmi:id="PartReference_1120070252273" part="TOKEN"
name="TOKEN">
            <transform xmi:id="Transform_1120070346871"
algorithm="http://www.w3.org/2002/06/xmldsig-filter2" name="token_transform">

```

```

        <properties xmi:id="Property_1120070500028"
name="com.ibm.wsspi.wssecurity.dsig.XPath2Expression_1"
value="/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and
local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/env
elope/' and
local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/0
1/oasis-200401-wss-wssecurity-secext-1.0.xsd' and
local-name()='Security']/*[namespace-uri()='urn:oasis:names:tc:SAML:1.0:asserti
on' and local-name()='Assertion']"/>
        <properties xmi:id="Property_1120070557490"
name="com.ibm.wsspi.wssecurity.dsig.XPath2Filter_1" value="intersect"/>
        <properties xmi:id="Property_1120070593456"
name="com.ibm.wsspi.wssecurity.dsig.XPath2Order_1" value="1"/>
    </transform>
    <transform xmi:id="Transform_1121201089711"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" name="token_transform_2"/>
    <digestMethod xmi:id="DigestMethod_1120070252280"
algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
</partReference>
    <signingKeyInfo xmi:id="SigningKeyInfo_1119714258006"
keyinfoRef="wssm_client_sig_keyinfo" name=""/>
</signingInfo>
    <encryptionInfo xmi:id="EncryptionInfo_1119714944732"
name="wssm_server_encinfo">
    <encryptionMethod xmi:id="DataEncryptionMethod_1119714944732"
algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
    <keyEncryptionMethod xmi:id="KeyEncryptionMethod_1119714944732"
algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <encryptionKeyInfo xmi:id="EncryptionKeyInfo_1119714944732"
keyinfoRef="wssm_server_enc_keyinfo" name=""/>
    <partReference xmi:id="PartReference_1119714944732"
part="BODY_AND_TOKEN" name=""/>
</encryptionInfo>
    <keyInfo xmi:id="KeyInfo_1119714213577" type="STRREF"
name="wssm_client_sig_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentGenerat
or">
    <keyLocatorMapping xmi:id="KeyLocatorMapping_1119714213577"
locatorRef="wssm_client_sig_keylocator"
keynameRef="cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us"/>
    <tokenReference xmi:id="TokenReference_1119714213577"
tokenRef="SigningToken" name=""/>
</keyInfo>
    <keyInfo xmi:id="KeyInfo_1119714893056" type="KEYID"
name="wssm_server_enc_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.KeyIdContentGenerator">
    <keyLocatorMapping xmi:id="KeyLocatorMapping_1119714893061"
locatorRef="wssm_server_enc_keylocator"
keynameRef="cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us"/>

```

```

        </keyInfo>
        <keyLocator xmi:id="KeyLocator_1119714078748"
name="wssm_client_sig_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
        <keyStore xmi:id="KeyStore_1119714078748" storepass="{xor}Lz4sLChvLTs="
path="rbtelco-signing.jks" type="JKS"/>
        <keys xmi:id="Key_1119714111348" alias="rbtelco_rbstocks"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us"/>
        </keyLocator>
        <keyLocator xmi:id="KeyLocator_1119714729644"
name="wssm_server_enc_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
        <keyStore xmi:id="KeyStore_1119714729644" storepass="{xor}Lz4sLChvLTs="
path="rbtelco-partners.jks" type="JKS"/>
        <keys xmi:id="Key_1119714800811" alias="rbstocks_rbtelco"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us"/>
        </keyLocator>
        <tokenGenerator xmi:id="TokenGenerator_1119713274158" name="SAML"
classname="com.tivoli.am.fim.wssm.tokengenerators.WSSTokenGenerator">
        <valueType xmi:id="ValueType_1119713274158" localName="Assertion"
uri="urn:oasis:names:tc:SAML:1.0:assertion" name=""/>
        <callbackHandler xmi:id="CallbackHandler_1119713285192"
classname="com.tivoli.am.fim.wssm.callbackhandlers.WSSMCallbackHandler"/>
        <properties xmi:id="Property_1120796292808" name="trust.service.call"
value="true"/>
        <properties xmi:id="Property_1120796345906" name="trust.service.url"
value="http://www.rbtelco.com:19082/TrustServer/SecurityTokenService"/>
        <properties xmi:id="Property_1120796361000" name="default.issuer.uri"
value="http://www.rbtelco.com/internal"/>
        <partReference xmi:id="PartReference_1119713274158" part="SAML"
name=""/>
        </tokenGenerator>
        <tokenGenerator xmi:id="TokenGenerator_1119713857718" name="SigningToken"
classname="com.ibm.wsspi.wssecurity.token.X509TokenGenerator">
        <valueType xmi:id="ValueType_1119715640357"
localName="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-p
rofile-1.0#X509" uri="" name=""/>
        <callbackHandler xmi:id="CallbackHandler_1119713895995"
classname="com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler">
        <key xmi:id="Key_1119713958371" alias="rbtelco_rbstocks"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us"/>
        <keyStore xmi:id="KeyStore_1119713895995"
storepass="{xor}Lz4sLChvLTs=" path="rbtelco-signing.jks" type="JKS"/>
        </callbackHandler>
        </tokenGenerator>

```

</securityRequestGeneratorBindingConfig>

Example: D-7 RBTelco Web services gateway client binding from RBStocks

```
<sibwssecurity:SIBWSSecurityResponseConsumerBindingConfig
xmi:id="SIBWSSecurityResponseConsumerBindingConfig_1119715281832"
name="RBStocks Response Consumer SigEnc">
  <securityResponseConsumerBindingConfig
xmi:id="SecurityResponseConsumerBindingConfig_1119715281832">
  <signingInfo xmi:id="SigningInfo_1119716031838"
name="wssm_server_signinfo">
  <signatureMethod xmi:id="SignatureMethod_1119716031838"
algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <canonicalizationMethod xmi:id="CanonicalizationMethod_1119716031838"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <partReference xmi:id="PartReference_1119716088225"
part="required_integrity" name="required_integrity">
  <transform xmi:id="Transform_1119716102736"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" name="body_transform"/>
  <digestMethod xmi:id="DigestMethod_1119716088225"
algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  </partReference>
  <signingKeyInfo xmi:id="SigningKeyInfo_1119716056223"
keyinfoRef="wssm_server_sig_keyinfo" name="wssm_server_sig_keyinfo_name"/>
  </signingInfo>
  <encryptionInfo xmi:id="EncryptionInfo_1119716155815"
name="wssm_client_encinfo">
  <encryptionMethod xmi:id="DataEncryptionMethod_1119716155815"
algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
  <keyEncryptionMethod xmi:id="KeyEncryptionMethod_1119716155815"
algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  <encryptionKeyInfo xmi:id="EncryptionKeyInfo_1119716927439"
keyinfoRef="wssm_client_enc_keyinfo" name="wssm_client_enc_keyinfo_name"/>
  <partReference xmi:id="PartReference_1119716155815"
part="required_confidentiality" name=""/>
  </encryptionInfo>
  <keyInfo xmi:id="KeyInfo_1119715979473" type="STRREF"
name="wssm_server_sig_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentConsumer">
  <keyLocatorMapping xmi:id="KeyLocatorMapping_1119715979473"
locatorRef="wssm_server_sig_keylocator" keynameRef=""/>
  <tokenReference xmi:id="TokenReference_1119715979473"
tokenRef="wssm_server_sig_consumer" name=""/>
  </keyInfo>
  <keyInfo xmi:id="KeyInfo_1119716000377" type="KEYID"
name="wssm_client_enc_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.KeyIdContentConsumer">
```

```

        <keyLocatorMapping xmi:id="KeyLocatorMapping_1119716000377"
locatorRef="wssm_client_enc_keylocator" keynameRef=""/>
        <tokenReference xmi:id="TokenReference_1119716000377"
tokenRef="wssm_client_enc_consumer" name=""/>
    </keyInfo>
    <keyLocator xmi:id="KeyLocator_1119715823397"
name="wssm_server_sig_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator"/>
    <keyLocator xmi:id="KeyLocator_1119715881978"
name="wssm_client_enc_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
        <keyStore xmi:id="KeyStore_1119715881985" storepass="{xor}Lz4sLChvLTs="
path="rbtelco-signing.jks" type="JKS"/>
        <keys xmi:id="Key_1119715931562" alias="rbtelco_rbstocks"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us"/>
    </keyLocator>
    <tokenConsumer xmi:id="TokenConsumer_1119715529546"
classname="com.ibm.wsspi.wssecurity.token.X509TokenConsumer"
name="wssm_server_sig_consumer">
        <valueType xmi:id="ValueType_1119715529546"
localName="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-p
rofile-1.0#X509" uri="" name=""/>
        <jAASConfig xmi:id="JAASConfig_1119715546365"
configName="system.wssecurity.X509BST"/>
        <certPathSettings xmi:id="CertPathSettings_1119715694465">
            <trustAnchorRef xmi:id="TrustAnchorRef_1119715694465"
ref="wssm_client_trust_anchor"/>
        </certPathSettings>
    </tokenConsumer>
    <tokenConsumer xmi:id="TokenConsumer_1119715758571"
classname="com.ibm.wsspi.wssecurity.token.X509TokenConsumer"
name="wssm_client_enc_consumer">
        <valueType xmi:id="ValueType_1119715758571"
localName="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-p
rofile-1.0#X509" uri="" name=""/>
        <jAASConfig xmi:id="JAASConfig_1119715769484"
configName="system.wssecurity.X509BST"/>
        <certPathSettings xmi:id="CertPathSettings_1119715758571">
            <trustAnyCertificate xmi:id="TrustAnyCertificate_1119715758571"/>
        </certPathSettings>
    </tokenConsumer>
    <trustAnchor xmi:id="TrustAnchor_1119715372375"
name="wssm_client_trust_anchor">
        <keyStore xmi:id="KeyStore_1119715372375" storepass="{xor}Lz4sLChvLTs="
path="rbtelco-ca.jks" type="JKS"/>
    </trustAnchor>
</securityResponseConsumerBindingConfig>

```

Web services server RBStocks

The Web services configuration at RBStocks receives the WS-Security message from the Web services gateway at RBTelco. It has signing and encryption over the security token (signed SAML assertion) and message body. The signed SAML assertion represents the client invoking the Web service. At RBStocks, the message is decrypted, and the signature over the token and body checked. Additionally, the signed SAML assertion is exchanged at the trust service for a SAML assertion representing the type of user (real-time, delayed, or blacklisted) for this invocation. More information on this configuration is contained in Chapter 10, “Use case 4 - Web services security management” on page 291.

RBStocks server extension configuration

Example D-8 shows the WS-Security server extension configuration for the stock quote Web services server at RBStocks.

Example: D-8 RBStocks WS-Security server extension

```
<com.ibm.etools.webservice.wsext:WsExtension xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:com.ibm.etools.webservice.wsext="http://www.ibm.com/websphere/appserver/s
chemas/5.0.2/wsext.xmi" xmi:id="WsExtension_1084457488394"
routerModuleName="WebProject.war">
  <wsDescExt xmi:id="WsDescExt_1119486860156"
wsDescNameLink="StockQuoteServiceService">
  <pcBinding xmi:id="PcBinding_1119711182734" pcNameLink="StockQuoteService"
scope="Session">
  <serverServiceConfig xmi:id="ServerServiceConfig_1119711182734">
  <securityRequestConsumerServiceConfig
xmi:id="SecurityRequestConsumerServiceConfig_1119711182734">
  <caller xmi:id="Caller_1120143005093" name="SAML" part=""
uri="urn:oasis:names:tc:SAML:1.0:assertion" localName="Assertion">
  <properties xmi:id="Property_1120143005109"
name="com.ibm.wsspi.wssecurity.caller.tokenConsumerNS"
value="urn:oasis:names:tc:SAML:1.0:assertion"/>
  <properties xmi:id="Property_1120143005110"
name="com.ibm.wsspi.wssecurity.caller.tokenConsumerLN" value="Assertion"/>
  </caller>
  <requiredSecurityToken xmi:id="RequiredSecurityToken_1120143005109"
name="SAML" uri="urn:oasis:names:tc:SAML:1.0:assertion" localName="Assertion"
usage="Required"/>
```

```

        <requiredConfidentiality
xmi:id="RequiredConfidentiality_1121303755781" name="required_confidentiality"
usage="Required">
        <messageParts xmi:id="MessageParts_1121303755781"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
keyword="bodycontent"/>
        <messageParts xmi:id="MessageParts_1121303755782"
Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116"
keyword="/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and
local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/env
elope/' and
local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/0
1/oasis-200401-wss-wssecurity-secext-1.0.xsd' and
local-name()='Security']/*[namespace-uri()='urn:oasis:names:tc:SAML:1.0:asserti
on' and local-name()='Assertion']"/>
        </requiredConfidentiality>
        <requiredIntegrity xmi:id="RequiredIntegrity_1120143005109"
name="required_integrity_body" usage="Required">
        <messageParts xmi:id="MessageParts_1120143005110"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
keyword="body"/>
        </requiredIntegrity>
        <requiredIntegrity xmi:id="RequiredIntegrity_1121280441031"
name="required_integrity_token" usage="Required">
        <messageParts xmi:id="MessageParts_1121280441031"
Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116"
keyword="/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and
local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/env
elope/' and
local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/0
1/oasis-200401-wss-wssecurity-secext-1.0.xsd' and
local-name()='Security']/*[namespace-uri()='urn:oasis:names:tc:SAML:1.0:asserti
on' and local-name()='Assertion']"/>
        </requiredIntegrity>
        </securityRequestConsumerServiceConfig>
        <securityResponseGeneratorServiceConfig
xmi:id="SecurityResponseGeneratorServiceConfig_1120143785750">
        <confidentiality xmi:id="Confidentiality_1120143785750" name="BODY"
order="2">
        <messageParts xmi:id="MessageParts_1120143785750"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
keyword="bodycontent"/>
        </confidentiality>
        <integrity xmi:id="Integrity_1120143785750" name="BODY" order="1">
        <messageParts xmi:id="MessageParts_1120143785751"
Dialect="http://www.ibm.com/websphere/webservices/wssecurity/dialect-was"
keyword="body"/>
        </integrity>
        </securityResponseGeneratorServiceConfig>

```

```

        </serverServiceConfig>
    </pcBinding>
</wsDescExt>
</com.ibm.etools.webservice.wsext:WsExtension>

```

RBStocks server binding configuration

Example D-9 shows the WS-Security server binding configuration for the stock quote Web services server at RBStocks. Note the special transform required in the SigningInformation. This has to match the transforms used in Example D-6 on page 443 to construct the signing information in the first place.

Example: D-9 RBStocks WS-Security server binding

```

<com.ibm.etools.webservice.wsbind:WSBinding xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:com.ibm.etools.webservice.wsbind="http://www.ibm.com/websphere/appserver/s
chemas/5.0.2/wsbind.xmi" xmi:id="WSBinding_1084457488514">
  <wsdescBindings xmi:id="WSDescBinding_1119486859984"
wsDescNameLink="StockQuoteServiceService">
    <pcBindings xmi:id="PCBinding_1119711182765" pcNameLink="StockQuoteService"
wsdlServiceQnameNamespaceLink="" wsdlServiceQnameLocalnameLink=""
scope="Session">
      <securityRequestConsumerBindingConfig
xmi:id="SecurityRequestConsumerBindingConfig_1119711182765">
        <signingInfo xmi:id="SigningInfo_1119721150171"
name="wssm_client_signinfo">
          <signatureMethod xmi:id="SignatureMethod_1120143507875"
algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <canonicalizationMethod xmi:id="CanonicalizationMethod_1120143507875"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <partReference xmi:id="PartReference_1119721150171"
part="required_integrity_body" name="">
            <transform xmi:id="Transform_1120143507875"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" name="body_transform"/>
            <digestMethod xmi:id="DigestMethod_1120147961484"
algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          </partReference>
          <partReference xmi:id="PartReference_1120143507875"
part="required_integrity_token" name="required_integrity_token">
            <transform xmi:id="Transform_1121280441062"
algorithm="http://www.w3.org/2002/06/xmldsig-filter2"
name="token_transform_1"/>
            <transform xmi:id="Transform_1121280441063"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" name="token_transform_2"/>
            <digestMethod xmi:id="DigestMethod_1120147961485"
algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

```



```

        </partReference>
        <signingKeyInfo xmi:id="SigningKeyInfo_1120143507875"
keyinfoRef="wssm_client_sig_keyinfo" name="wssm_client_sig_keyinfo_name"/>
    </signingInfo>
    <encryptionInfo xmi:id="EncryptionInfo_1120143785781"
name="wssm_server_encinfo">
        <encryptionMethod xmi:id="DataEncryptionMethod_1120143785781"
algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <keyEncryptionMethod xmi:id="KeyEncryptionMethod_1120143785781"
algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <encryptionKeyInfo xmi:id="EncryptionKeyInfo_1120143785781"
keyinfoRef="wssm_server_enc_keyinfo" name="wssm_server_enc_keyinfo_name"/>
        <partReference xmi:id="PartReference_1120143785781"
part="required_confidentiality"/>
    </encryptionInfo>
<keyInfo xmi:id="KeyInfo_1120143507875" type="STRREF"
name="wssm_client_sig_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentConsumer">
    <keyLocatorMapping xmi:id="KeyLocatorMapping_1120143507875"
locatorRef="wssm_client_sig_keylocator" keynameRef=""/>
    <tokenReference xmi:id="TokenReference_1120143507875"
tokenRef="wssm_client_sig_consumer"/>
</keyInfo>
    <keyInfo xmi:id="KeyInfo_1120143507876" type="KEYID"
name="wssm_server_enc_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.KeyIdContentConsumer">
        <keyLocatorMapping xmi:id="KeyLocatorMapping_1120143507876"
locatorRef="wssm_server_enc_keylocator" keynameRef=""/>
        <tokenReference xmi:id="TokenReference_1120143507876"
tokenRef="wssm_server_enc_consumer"/>
    </keyInfo>
    <keyLocator xmi:id="KeyLocator_1119721150171"
name="wssm_client_sig_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator"/>
        <keyLocator xmi:id="KeyLocator_1120143507875"
name="wssm_server_enc_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
            <keyStore xmi:id="KeyStore_1120143507875"
storepass="{xor}Lz4sLChvLTs=" path="rbstocks-signing.jks" type="JKS"/>
            <keys xmi:id="Key_1120143507875" alias="rbstocks_rbtelco"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us"/>
        </keyLocator>
        <tokenConsumer xmi:id="TokenConsumer_1120143167906"
classname="com.tivoli.am.fim.wssm.tokenconsumers.WSSMTokenConsumer"
name="SAML">
            <valueType xmi:id="ValueType_1120143167906" localName="Assertion"
uri="urn:oasis:names:tc:SAML:1.0:assertion" name=""/>

```

```

        <jAASConfig xmi:id="JAASConfig_1120143167906"
configName="system.itfim.wssm.samla"/>
        <properties xmi:id="Property_1120143167906" name="trust.service.call"
value="true"/>
        <properties xmi:id="Property_1120143167907" name="trust.service.url"
value="http://www.rbstocks.com:9082/TrustServer/SecurityTokenService"/>
        <partReference xmi:id="PartReference_1120143167906" part="SAML"/>
    </tokenConsumer>
    <tokenConsumer xmi:id="TokenConsumer_1120143167907"
classname="com.ibm.wsspi.wssecurity.token.X509TokenConsumer"
name="wssm_client_sig_consumer">
        <valueType xmi:id="ValueType_1120143167907"
localName="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-p
rofile-1.0#X509" uri="" name="wssm_client_sig_consumer_vtype"/>
        <jAASConfig xmi:id="JAASConfig_1120143167907"
configName="system.wssecurity.X509BST"/>
        <partReference xmi:id="PartReference_1120143167907"/>
        <certPathSettings xmi:id="CertPathSettings_1120143167906">
            <trustAnchorRef xmi:id="TrustAnchorRef_1120143167906"
ref="wssm_server_trust_anchor"/>
        </certPathSettings>
    </tokenConsumer>
<tokenConsumer xmi:id="TokenConsumer_1120143167908"
classname="com.ibm.wsspi.wssecurity.token.X509TokenConsumer"
name="wssm_server_enc_consumer">
        <valueType xmi:id="ValueType_1120143167908"
localName="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-p
rofile-1.0#X509" uri="" name="wssm_server_enc_consumer_vtype"/>
        <jAASConfig xmi:id="JAASConfig_1120143167908"
configName="system.wssecurity.X509BST"/>
        <partReference xmi:id="PartReference_1120143167908"/>
        <certPathSettings xmi:id="CertPathSettings_1120143167907">
            <trustAnchorRef xmi:id="TrustAnchorRef_1120143167907"
ref="wssm_server_trust_anchor"/>
        </certPathSettings>
    </tokenConsumer>
    <trustAnchor xmi:id="TrustAnchor_1120143167906"
name="wssm_server_trust_anchor">
        <keyStore xmi:id="KeyStore_1120143167906"
storepass="{xor}Lz4sLChvLTs" path="rbstocks-ca.jks" type="JKS"/>
    </trustAnchor>
</securityRequestConsumerBindingConfig>
<securityResponseGeneratorBindingConfig
xmi:id="SecurityResponseGeneratorBindingConfig_1120143785781">
    <signingInfo xmi:id="SigningInfo_1120144058984"
name="wssm_server_signinfo">
        <signatureMethod xmi:id="SignatureMethod_1120144058984"
algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

```

```

    <canonicalizationMethod xmi:id="CanonicalizationMethod_1120144058984"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <partReference xmi:id="PartReference_1120144058984" part="BODY"
name="">
        <transform xmi:id="Transform_1120144058984"
algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" name="body_transform"/>
        <digestMethod xmi:id="DigestMethod_1120144058984"
algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    </partReference>
    <signingKeyInfo xmi:id="SigningKeyInfo_1120144058984"
keyinfoRef="wssm_server_sig_keyinfo" name="wssm_server_sig_keyinfo_name"/>
    </signingInfo>
    <encryptionInfo xmi:id="EncryptionInfo_1120144058984"
name="wssm_client_encinfo">
        <encryptionMethod xmi:id="DataEncryptionMethod_1120144058984"
algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <keyEncryptionMethod xmi:id="KeyEncryptionMethod_1120144058984"
algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <encryptionKeyInfo xmi:id="EncryptionKeyInfo_1120144058984"
keyinfoRef="wssm_client_enc_keyinfo" name="wssm_client_enc_keyinfo_name"/>
        <partReference xmi:id="PartReference_1120144058985" part="BODY"/>
    </encryptionInfo>
    <keyLocator xmi:id="KeyLocator_1120143785781"
name="wssm_server_sig_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
        <keyStore xmi:id="KeyStore_1120143785781"
storepass="{xor}Lz4sLChvLTs=" path="rbstocks-signing.jks" type="JKS"/>
        <keys xmi:id="Key_1120143785781" alias="rbstocks_rbtelco"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us"/>
    </keyLocator>
    <keyLocator xmi:id="KeyLocator_1120144058984"
name="wssm_client_enc_keylocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
        <keyStore xmi:id="KeyStore_1120144058984" storepass="{xor}Lz4sLChvLTs="
path="rbstocks-partners.jks" type="JKS"/>
        <keys xmi:id="Key_1120144058984" alias="rbtelco_rbstocks"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us"/>
    </keyLocator>
    <keyInfo xmi:id="KeyInfo_1120144058984" type="STRREF"
name="wssm_server_sig_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentGenerat
or">
        <keyLocatorMapping xmi:id="KeyLocatorMapping_1120144058984"
locatorRef="wssm_server_sig_keylocator"
keynameRef="cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us"/>
        <tokenReference xmi:id="TokenReference_1120144058984"
tokenRef="SigningToken"/>

```

```

        </keyInfo>
        <keyInfo xmi:id="KeyInfo_1120144058985" type="KEYID"
name="wssm_client_enc_keyinfo"
classname="com.ibm.ws.webservices.wssecurity.keyinfo.KeyIdContentGenerator">
        <keyLocatorMapping xmi:id="KeyLocatorMapping_1120144058985"
locatorRef="wssm_client_enc_keylocator"
keynameRef="cn=rbtelco_rbstocks.rbtelco.com,o=rbtelco,c=us"/>
        </keyInfo>
        <tokenGenerator xmi:id="TokenGenerator_1120143785781"
name="SigningToken"
classname="com.ibm.wsspi.wssecurity.token.X509TokenGenerator">
        <valueType xmi:id="ValueType_1120143785781"
localName="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-p
rofile-1.0#X509" uri="" name="SigningToken_vtype"/>
        <callbackHandler xmi:id="CallbackHandler_1120143785781"
classname="com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler">
        <key xmi:id="Key_1120143785782" alias="rbstocks_rbtelco"
keypass="{xor}Lz4sLChvLTs="
name="cn=rbstocks_rbtelco.rbstocks.com,o=rbstocks,c=us"/>
        <keyStore xmi:id="KeyStore_1120143785782"
storepass="{xor}Lz4sLChvLTs=" path="rbstocks-signing.jks" type="JKS"/>
        <basicAuth xmi:id="BasicAuth_1120143785781"/>
        </callbackHandler>
        <partReference xmi:id="PartReference_1120143785782"/>
        </tokenGenerator>
    </securityResponseGeneratorBindingConfig>
</pcBindings>
</wsdescBindings>
</com.ibm.etools.webservice.wsbind:WSBinding>

```

Glossary

Access Control Lists (ACL) A cornerstone of security is the ability to determine who can access computer networks and systems. Control can be exercised through the use of access control protocols, computer applications that authenticate the user logging into a network. Access Control Lists define which users can access specific data and programs. Access codes are passwords, series of characters or numbers that enable a user to access the network.

Active Requestors An application (possibly a Web browser) that is capable of issuing Web services messages such as those described in WS-Security and WS-Trust.

AEF Access Enforcement Point.

Agent A function that represents a requester to a server. An agent can be present in both a source and a target system.

Application Programming Interface

API Software applications, such as spreadsheets or word processing, use a special language and message format—the API—to communicate with the computer operating system, database management system, or other system programs.

Assertion In computer programming, an assertion is a programming language construct that immediately aborts program execution if a certain condition or expression is false (an *assertion failure*). It is used by programmers during development to check for potential errors or bugs. To assist with this, the implementation of assertions in many languages provides information such as the file name and line number in the source code that triggered the assertion failure.

Association The process by which principals become associated or affiliated with a trust realm or federations.

Assurance Assurance is the determination that host platforms, end-user platforms, applications, network component configurations, and operations are in accordance with security policy. Entities are monitored to ensure policies have been implemented and used. Detected noncompliance with policies is recorded and reported. Remediation of policy noncompliance is based on remediation policy.

Asymmetric Keys In computer security, the two keys in a key pair. The keys are called asymmetric because one key holds more of the encryption pattern than the other does.

Attribute Service A Web service that maintains information (attributes) about principals within a trust realm or federation. The term principal, in this context, can be applied to any system entity, not just a person.

Audit The recording of security events in a log. To ensure future claims that security events recorded are accurate and have not been altered (that is, are non-reputable), audit records are collected and secured. Audit records may be used for:

- ▶ Internal problem analysis
- ▶ Use as evidence in relation to a potential breach of contract, breach of regulatory requirement, or in the event of civil or criminal proceedings, for example, under computer misuse or data protection legislation
- ▶ Negotiating for compensation from software and service suppliers

Audit logs are created by system components, including operating systems, applications, and network devices.

Authentication Authentication denotes a security procedure where an individual is identified. The process ensures that the individual is whom he or she claims to be, but does not affect the individual's access rights. User names, passwords, and biometric scanning are all authentication techniques.

Authorization This phase of security admits only legitimate user access to systems, data, applications or networks. After the user is authenticated, he is authorized, that is, granted access to a network resource. An identification number or password that is used to gain access to a local or remote computer system.

Anti-Virus Management Anti-virus (AV) clients run on host platforms. Anti-virus management includes the following:

- ▶ AV client distribution and updates to authorized platforms. Platforms may be initially loaded with AV clients or fetch AV clients from the AV manager.
- ▶ Notification that updates are available.
- ▶ Making AV clients and updates available for automatic download when the host platform connects to the manager.
- ▶ Receiving host AV log files and host AV configuration data.
- ▶ Providing summary AV event information and alerts.
- ▶ Providing reports.

B2B Business to Business.

B2C Business to Consumer.

B2E Business to Employee.

Binding Security and Secure

Conversation Security binding is the protocol that ties security attributes together, such as an identity and the authorizations for the identity. Examples of security bindings are:

- ▶ Secure Sockets Layer and Transport Layer Security protocols provide for the secure authentication of servers and clients.
- ▶ X.509 certificates bind an identity to a public key.
- ▶ A Web cookie binds an identity to a service.

Security conversion securely maps information from one form to another form. For example, a password and ID may be converted to a common format for an authenticated identity. Confidentiality may convert plain text information into cipher text using an encryption key or keys.

CDC Common Domain Cookies.

Certificate The most common kind of credential in the network computing environment. Certificates include standard information such as the owner's public key, globally accessible name, and expiration dates; certificates may also contain some application-unique data such as title, degree(s) earned, and professional licenses. Certificates are also called digital certificates.

Certificate Authority (CA) In the pre-Internet world, every secure transaction involved a trusted third party—such as a notary, attorney or broker—who could guarantee that both parties were who they purported to be. A Certificate Authority fills that same role in the digital world. A CA vendor, such as VeriSign or Entrust, issues certificates that contain the identities and affiliations of individuals, along with their public keys. These certificates are bound together with the digital signature and stored in a special directory. The sender's browser looks up the recipient's certificate in the directory, and the message can be encrypted using the key embedded in the certificate. The sender can then sign the message using his own private key, and the recipient can verify the signature by using the sender's public key that is vouched for by the CA.

CGI (Common Gateway Interface) A specification for transferring information between a World Wide Web server and a CGI program. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, Java, or Visual Basic. CGI programs are the most common way for Web servers to interact dynamically with users. Many HTML pages that contain forms, for example, use a CGI program to process the form's data once it is submitted.

Circle of Trust The group of service providers that share linked identities and have business and operating agreements in place is known as a circle of trust.

Claim A declaration made by an entity (for example, name, identity, key, group, privilege, capability, attribute, and so on).

Claim Confirmation The process of verifying that a claim applies to an entity.

Common Object Request Broker Architecture (CORBA) An architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker." CORBA was developed by a consortium of vendors through the Object Management Group, which currently includes over 500 member companies. Both the International Organization for Standardization (ISO) and X/Open have sanctioned CORBA as the standard architecture for distributed objects (which are also known as components). CORBA 3 is the latest level.

Container A Java run-time environment for enterprise beans. A container, which runs on an Enterprise JavaBeans server, manages the life cycles of enterprise bean objects, coordinates distributed transactions, and implements object security.

Credential Exchange The purpose of a credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms, the processes, and the security subsystems within a computing solution. Credentials are created as a result of a successful authentication. Some common types of credentials are:

- ▶ X.509 public key identity certificates that bind an identity to a public key.
- ▶ X.509 attribute certificates that bind an identity or a public key with some attribute.

Kerberos tickets that are encrypted messages binding the holder with some attribute or privilege, and encrypted cookies.

Credentials Data associated with a user or resource that indicates identity and authority level. Credentials need to be issued by a trustworthy authority, as that authority is vouching for the identity and authorization level. A passport is a credential; it represents the bearer's identity and rights and is issued by a formally recognized government agency. In network computing environments, the most common type of credential is a certificate that has been created and "signed" by a trusted Certificate Authority.

CUID Common Unique Identifier.

Demilitarized Zone (DMZ) An area of your network that separates it from other areas of the network, including the Internet.

Digital Certificate Digital certificates allow a user to send an encrypted message. A digital certificate is an attachment to an electronic message that verifies the user is who one claims to be, and is used to ensure secure e-business transactions. The Certificate Authority (CA), which issues a user's digital certificate, makes known the user's public key, which another user employs to decode the digital certificate attached to a message. This process also verifies that the certificate was issued by the CA and allows users to obtain identification information of the certificate-holding sender. The recipient of the message can then send an encrypted reply.

Directory A directory service is the "yellow pages" of computer network resources, stored on a server and often containing security-related data, such as phone numbers, e-mail addresses, public keys, computer names, and addresses. The data is presented hierarchically, much like a family tree, with one section providing key information about the files beneath it. To access a file, a user may need to produce the names of all the directories above it by specifying a path. To read information from or write information into a directory, the user must use operating system commands.

Directory Services Provide means of locating resources and users in a network or networks. They are analogous to telephone directories—even though you look up a resource or user name, you still need to know something about its location to narrow the search. A directory can also include the public key of the user or resource in addition to location and other information.

Domain or Realm A domain or realm represents a single unit of security administration or trust.

EAI WebSEAL External Authentication Interface.

Enterprise JavaBeans (EJB) An architecture for setting up program components, written in the Java programming language, that run in the server parts of a computer network that uses the client/server model. Enterprise Java Beans is built on the JavaBeans technology for distributing program components to clients in a network. Enterprise Java Beans offer enterprises the advantage of being able to control change at the server rather than having to update each individual computer with a client whenever a new program component is changed or added. EJB components have the advantage of being reusable in multiple applications. To deploy an EJB Bean or component, it must be part of a specific application, which is called a container.

Enterprise Service Bus (ESB) is an emerging standard for integrating enterprise applications in an implementation-independent fashion, at a coarse-grained service level (leveraging the principles of service-oriented architecture) via an event-driven and XML-based messaging engine (the bus).

Federation A group of two or more organizations that have agreed to allow a user from one federation partner to seamlessly access resources from another partner in a secure and trustworthy manner.

FIM Federated Identity Management/Manager.

Firewall A firewall is a hardware/software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets, and can block some virus attacks—as long as those viruses are coming from the Internet.

FTN Liberty Federation Termination Identification.

FULM Federated User life cycle Management.

Generic Security Services Application Program Interface (GSS-API) is defined in RFC 2853. GSS-API offers application programmers uniform access to security services atop a variety of underlying security mechanisms, including Kerberos.

HTTP point of contact (PoC) A generic component normally located in a DMZ. It is typically an HTTP reverse proxy, or similar component, capable of authenticating a user and managing a session for that user.

Intrusion Defense Intrusion Defense provides defense against attackers attempting to gain access to a network, device or host. Intrusion detection and response capabilities monitor network segments and hosts within a centralized operational and management framework. Responses to detected intrusion attempts include inputs to event management systems, paging, and trouble ticket systems. Intrusion defense is installed on hosts, desktops, laptops, and on-network devices. Intrusion Defense management includes the life cycle management of intrusion detection mechanisms on hosts, desktops, and laptops and on network devices:

- ▶ ID application distribution and updates to authorized platforms. Host platforms may be initially loaded with ID clients or fetch ID clients from the ID manager.
- ▶ Notification that ID updates are available.
- ▶ Making ID clients and updates available for automatic download when the host platform connects to the manager.
- ▶ Receiving host ID security event logs and performance log files and host ID configuration data.
- ▶ Providing summary ID event information and alerts.
- ▶ Providing reports.

Identity Mapping A method of creating relationships between identity properties. Some identity providers may make use of identity mapping.

Identity provider (IdP) An entity that acts as a peer entity authentication service to end requestors and data origin authentication service to service providers (this is typically an extension of a security token service).

IE Internet Explorer.

Identity Management In accordance with document security policy, identity management includes the

- ▶ Identity proofing, identity approval, and identity rights authorization.
- ▶ Identity token creation and token distribution to the user.
- ▶ (Dynamically) provisioning user identity, rights, and profile to relying parties (operating systems, and applications).
- ▶ User profile management.
- ▶ Enabling user self-care.
- ▶ Delegate administrative responsibility for approval and authorization as needed.
- ▶ Processes for token changes IAW policy, revoking, and approving reissue of new/changed token.
- ▶ Performing identity management in accordance with security policy.

IMS Identity Management System.

Internet Inter-ORB Protocol (IIOP) A protocol developed by the Object Management Group (OMG) to implement CORBA solutions over the World Wide Web. IIOP enables browsers and servers to exchange integers, arrays, and more complex objects, unlike HTTP, which only supports transmission of text.

IPI Identity provider Introduction.

ISC Integrated Systems Console.

Java 2 Platform Enterprise Edition (J2EE) A Java platform designed for the mainframe-scale computing typical of large enterprises. Sun Microsystems, together with industry partners such as IBM, designed J2EE to simplify application development in a thin client-tiered environment.

Java Database Connectivity (JDBC) An application program interface (API) specification for connecting programs written in Java to the data in popular database. The application program interface lets you encode access request statements in structured query language (SQL) that are then passed to the program that manages the database. It returns the results through a similar interface. JDBC is very similar to the SQL Access Group's Open Database Connectivity (ODBC); and, with a small "bridge" program, you can use the JDBC interface to access databases through the ODBC interface.

Java Naming and Directory Interface (JNDI) Enables Java platform-based applications to access multiple naming and directory services. Part of the Java Enterprise application programming interface (API) set, JNDI makes it possible for developers to create portable applications that are enabled for a number of different naming and directory services, including file systems, directory services, such as Lightweight Directory Access Protocol (LDAP), Novell Directory Services, and Network Information System (NIS); and distributed object systems, such as the Common Object Request Broker Architecture (CORBA), Java Remote Method Invocation (RMI), and Enterprise JavaBeans (EJB).

Java Security Specific security protocols are launched to protect programs using Java, a computer programming language mostly used for the World Wide Web. Java programs, which can be downloaded from a Web server and run on Java-compatible browsers, are run in a small, constrained area called a Sandbox. The Sandbox contains a security system that checks and verifies all codes coming into it. Java Security employs data encryption, where keys are needed to encrypt and read data.

Java Server Page (JSP) A technology for controlling the content or appearance of Web pages through the use of servlets, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it.

JAX-RPC A specification that describes application programming interfaces (APIs) and conventions for building Web services and Web service clients that used remote procedure calls (RPC) and XML. JAX-RPC is also known as JSR 101.

JKS Java Key Store.

Kerberos A network authentication protocol developed at the Massachusetts Institute of Technology (MIT). It is designed to provide strong authentication for client/server applications across insecure network connections by using secret-key cryptography.

Key Escrow The storing of a key (or parts of a key) with a trusted party or trusted parties in case of loss or destruction of the key.

Key management In accordance with document policy, key management provides life cycle management for public-private key pairs using a trusted Public key Infrastructure (enterprise or outsourced) operating in accordance with a documented Certificate Policy. Private keys and X.509 certificates can be used to provide authentication, confidentiality, data integrity, and non-repudiation for transactions and other data.

Key Recovery A process used to recover encrypted information that does not involve the storing of the key or any part of the key with a third party. Sometimes, important data needs to be recovered without normal access. The encryption key may have been lost accidentally, or an organization may need to audit its resources, or the data may be needed by law enforcement and other outside authorities. Key-recovery systems, like those proposed by National Institute for Standards and Technology (NIST), rely on close cooperation between certification authorities and user communities that share a public-key infrastructure (PKI). These groups would need to share components of encryption keys that are stored at separate locations. Many organizations find key recovery a preferable process to key escrow. The US government recently relaxed controls on the export of strong encryption based upon the development of key recovery technology by the computer industry.

LECP Liberty-enabled Client/Proxy.

Liberty Alliance is a consortium 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.

Lightweight Directory Access Protocol (LDAP) A software protocol for enabling anyone to locate organizations, individuals, and other resources (such as files and devices) in a network, whether on the public Internet or on a corporate intranet. LDAP is a "lightweight" (smaller amount of code) version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network.

Lightweight Third Party Authentication (LTPA) Implements an authentication protocol that uses a trusted third-party Lightweight Directory Access Protocol (LDAP) server. LTPA causes a search to be performed against the LDAP directory. LTPA supports both the basic and certificate challenge type.

Mapping Rules Rules used to convert a security item from form understood by an origin process to a form understood by a destination process. For example, an application can authenticate a user via any mechanism it chooses (ID/password, certificate, and so on), and then based on the mapping rules convert the authenticated identity to an identity format defined for a directory.

MASS Method for Architecting Secure Solutions.

Mobile Station International ISDN Number (MSISDN) The standard international telephone number used to identify a given subscriber. The number is based on the ITU-T (International Telecommunications Union-Telecommunication Standardization Sector) E.164 standard.

Network Security Solutions Network security solutions for on demand provide secure connectivity and access control to and for the enterprise network. Remote connections to the enterprise network can use a variety of technologies such as dialup and Virtual Private Network (SSL and IPSEC). Network firewalls permit only connections that are specified, in directions that are specified, and using protocols that are specified. Network security solutions feature centralized managed, log, and security event audit trail generation and collection, and report generation.

Non-repudiation Non-repudiation occurs when a document or participant in an activity is valid. In digital cryptography, this applies to a person who uses a private key to protect access. This guarantees that any messages signed using that person's digital signature could only have come from them. In e-commerce, when the key holder uses a digital signature in a financial transaction, it guarantees that the person making the transaction is who they claim to be.

OASIS (Organization for the Advancement of Structured Information Standards) is a global consortium that drives the development of e-business and Web service standards.

On Demand Operating Environment (ODOE)

The new computing architecture designed to help companies realize the benefits of on demand business. The on demand operating environment has four essential characteristics: It is integrated, open, virtualized, and autonomic.

Open Platform for Security Check Point

(OPSEC) The initiative to provide a common architecture for integrating security solutions.

Passive Requestor An HTTP browser capable of broadly supported HTTP (for example, HTTP/1.1).

PEP Policy Enforcement Point.

PKI A public key infrastructure enables users of a basically unsecure public network, such as the Internet, to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority.

Point of contact (PoC) A generic component, normally located in the DMZ. It is typically an HTTP reverse proxy, or similar component, capable of authenticating a user and managing a session for that user. Typically the PoC will have a connection to a local user registry, used to validate user authentication credentials presented by the user and also to retrieve user attributes/privilege information used with session management for an authenticated user.

Policy Management Policy management in the On Demand Security Infrastructure is the consistent application of enterprise security policy to on demand infrastructure components, services, and applications; network security solutions; and on demand security infrastructure components and services. Policy management is applied independent of application logic and operating system platform and includes trusted identity and token life cycle management identity, access control/authorization life cycle management, federated identity life cycle, privacy, single sign on, compliance determination and remediation, security event auditing and processing, and failure situations.

Portal A term, generally synonymous with gateway, for a World Wide Web site that is a major starting site for users when they get connected to the Web or that users tend to visit as an anchor site, linking to many other sites. Typical services offered by portal sites include a directory of Web sites, the ability to search for information, news, weather information, e-mail, stock quotes, phone and map information, and sometimes a community forum. Excite is among the first portals to offer users the ability to personalize that Web site according to individual interests.

Privacy Policies Security policies for managing access to and use of sensitive personal information, referred to as privacy-sensitive information. Individuals who provide personal information, such as social security numbers, have the right to determine when, how, and to what extent their personal information is used by organizations that collect the information.

Profile A document that describes how this model is applied to a specific class of requestor (for example, passive or active)

Proxy An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are served internally or by passing them, with possible translation, on to other servers. A proxy must interpret and, if necessary, rewrite a request message before forwarding it. Proxies are often used as client-side portals through network firewalls and as helper applications for handling requests via protocols not implemented by the user agent.

Pseudonym Service A Web service that maintains alternate identity information about principals within a trust realm or federation. The term principal, in this context, can be applied to any system entity, not just a person.

Public Key In asymmetric cryptography, the key that is made available for others to use to encrypt information. The owner of the associated private key is the only person who can decrypt the information.

Public Key Infrastructure (PKI) PKI is a system for verifying the authenticity of each party involved in an Internet transaction, protecting against fraud or sabotage, and for non-repudiation purposes so that consumers and retailers may protect themselves against denial of transactions. Trusted third-party organizations called certificate authorities issue digital certificates—attachments to electronic messages—that specify key components of the user's identity. During an Internet transaction signed, encrypted messages from one party to another are automatically routed to the Certificate Authority, where the certificates are verified before the transaction can proceed. PKI can be embedded in software applications, or offered as a service or a product. e-business leaders agree that PKIs are critical for transaction security and integrity, and the software industry is moving to adopt open standards for their use. Simplifying the directory systems that contain PKI data remains a challenge.

RA A Registration Authority is an authority in a network that verifies user requests for a digital certificate and tells the Certificate Authority (CA) to issue it. RAs are part of a public key infrastructure (PKI), a networked system that enables companies and users to exchange information and money safely and securely. The digital certificate contains a public key that is used to encrypt and decrypt messages and digital signatures.

Realm or Domain A realm or domain represents a single unit of security administration or trust.

Remote Method Invocation (RMI) This is the standard specification of the Java RPC.

Role-Based Access Control (RBAC) A method of granting access rights to users based on their assignment to a defined role in the organization.

Router An interconnection device that links two discrete networks and forwards packets between them. A router uses a networking protocol such as IP to address and direct data packets flowing into and out of the network on which it sits.

Secure Logging Secure logging is the means of recording security events and the protection provided to such logs to ensure their non-repudiation. Secure logging also includes a means for processing logs and generating reporting.

Secure Networks and Operating Systems Secure networks are networks that have implemented logical and physical access controls and may have implemented confidentiality, data integrity, and non-repudiation security services to restrict data access and network management to authorized personnel or entities. Secure operating systems are operating systems that have implemented logical and physical access controls and may have implemented confidentiality, data integrity, and non-repudiation security services to restrict data access and network management to authorized personnel or entities. Secure networks and operating systems generate security event audit records and are securely managed.

Secure Sockets Layer (SSL) A commonly used protocol for managing the security of a message transmission on the Internet. SSL has recently been succeeded by Transport Layer Security (TLS), which is based on SSL.

Security Assertion Markup Language (SAML) A specification designed to provide cross-vendor single sign-on interoperability.

Security Policy Expression Security policy expression is the means by which security policy is applied to or implemented for specific IT system components and applications. For example, firewall filtering rules in a file, hardware settings, and network configurations.

Security token Represents a collection of claims.

Security token service (STS) A Web service that issues security tokens. That is, it makes assertions based on evidence that it trusts, whoever trusts it. To communicate trust, a service requires proof, such as a security token or set of security tokens, and issues as security token with its own trust statement (note that for some security token formats this can just be a reassurance or co-signature). This forms the basis of trust brokering.

Service Oriented Architecture (SOA) expresses a software architectural concept that defines the use of services to support the requirements of software users. In a SOA environment, nodes on a network make resources available to other participants in the network as independent services that the participants access in a standardized way.

Service/endpoint policy Corporate security policy applied to or developed for services and information technology endpoints including response to legal, regulatory, and legislative requirements. Service policy states the specific security requirements for a service that generally is provided by a configuration of hosts, networks components, and applications. Endpoint policy states the specific security configuration to be implemented an individual host, network component, or application, and the protocols used to implement the service policy.

Signature A value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the signature to verify that the data has not been altered since it was signed by the signer.

Signed security token A security token that is asserted and cryptographically signed by a specific authority (for example, an X.509 certificate or a Kerberos ticket).

Sign-in The process by which security tokens are obtained for realm/domain or federation.

Sign-out The process by which security tokens are destroyed for realm/domain or federation.

Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) A mechanism that allows the secure negotiation of the mechanism to be used by two different GSS-API implementations. In essence, SPNEGO defines a universal but separate mechanism, solely for the purpose of negotiating the use of other security mechanisms. SPNEGO itself does not define or provide authentication or data protection, although it can allow negotiators to determine if the negotiation has been subverted, once a mechanism is established.

Simple Authentication and Security Layer (SASL) SASL (defined by RFC 2222) is a generic protocol framework that provides the means to use authentication mechanisms other than simple authentication and SSL over connection-based protocols. Protocols such as LDAP, POP, IMAP, and SMTP specify a SASL profile, which describes how to encapsulate SASL negotiation and SASL messages for the protocol. Within the SASL framework, different authentication schemes are referred to as mechanisms.

Simple Object Access Protocol (SOAP) A way for a program running in one kind of operating system to communicate with a program in the same or another kind of an operating system by using the HTTP Protocol and XML as the mechanisms for information exchange.

Single Sign-on (SSO) An optimization of the authentication sequence to remove the burden of repeating actions placed on the requestor. To facilitate SSO, an element called an Identity provider can act as a proxy on a requestor's behalf to provide evidence of authentication events to third parties requesting information about the requestor. These identity providers (IPs) are trusted third parties and need to be trusted by both the requestor (to maintain the requestor's identity information, as the loss of this information can result in the compromise of the requestor's identity) and the Web services that may grant access to valuable resources and information based upon the integrity of the identity information provided by the IP.

SLO Liberty Single Sign-Out.

Smart card A smart card is a small device the size of a credit card with built-in electronic memory of personal data, such as identification and financial information.

SP Service provider.

SPS SSO Protocol Services.

Stateful Packet Inspection (SPI) A firewall technology that examines the content of packets to determine whether they will be given access to a network.

Switch A hardware device that serves as a central connection point for all network cables. In a relatively small networking environment, a switch of four to 12 ports may be part of a router or gateway.

TAM IBM Tivoli Access Manager.

TDS IBM Tivoli Directory Server.

TFIM Tivoli Federated Identity Manager.

TIM IBM Tivoli Identity Manager.

Transport Layer Security (TLS) A protocol that ensures privacy between communicating applications and their users on the Internet. When a server and client communicate, TLS ensures that no third party may eavesdrop or tamper with any messages. TLS is the successor to the Secure Sockets Layer Protocol (SSL).

Trust According to the ITU-T X.509, Section 3.3.54, trust is defined as follows: "Generally an entity can be said to trust a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects".

Trust Domain An administered security space in which the source and target of a request can determine and agree whether particular sets of credentials from a source satisfy the relevant security policies of the target. The target may defer the trust decision to a third party, thus including the trusted third party in the Trust Domain.

Trust Modeling A trust model is a description/definition of how trust is established or conveyed between two entities or among multiple entities that operate under a common set of security policies.

Trusted Third Party A mechanism in which a trusted party creates a key and then keeps a copy of it in case of loss or destruction of the key, or legitimate request from law enforcement.

Uniform Resource Identifier (URI) The way you identify any point of content, whether it be a page of text, a video or sound clip, a still or animated image, or a program. The most common form of URI is the Web page address, which is a particular form or subset of URI called a Uniform Resource Locator (URL)

Uniform Resource Locator (URL) The unique address for a file that is accessible on the Internet. A common way to get to a Web site is to enter the URL of its home page file in your Web browser's address line.

Universal Description, Discovery and Integration (UDDI) Describes how a Registry can be used to publish and discover information about businesses and the Web Services they support.

Validation Service A Web service that uses the WS-Trust mechanisms to validate provided tokens and assess their level of trust (for example, claims trusted).

Virtual Organization Policies A statement of security policies for an IT system supporting the business needs of a specific subset of an enterprise or an IT system supporting cross-enterprise business needs operating under a common objective.

WAYF Where are you from.

Web services A way of providing computational capabilities using standard Internet protocols and architectural elements. For example, a database Web service would use Web browser interactions to retrieve and update data located remotely.

Web Services Description Language (WSDL) An XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically. WSDL is the cornerstone of the Universal Description, Discovery, and Integration (UDDI) initiative spearheaded by Microsoft, IBM, and Ariba.

Web Services Policy (WS-Policy) Provides a general purpose model and syntax to describe and communicate the policies of a Web service.

Web Services Security (WS-Security) A mechanism for incorporating security information into SOAP messages. While SOAP provides a flexible technique for structuring messages, it does not directly address how to secure these messages. WS-Security builds from the SOAP specification, structuring the use of essential security capabilities. Specifically, WS-Security uses binary tokens for authentication, digital signatures for integrity, and content-level encryption for confidentiality. By structuring SOAP security, WS-Security makes it easy to include security elements into SOAP through tools and enterprise applications.

Web Services Trust (WS-Trust) Describes a framework for trust models that enables Web services to securely interoperate.

Wireless Application Protocol (WAP) A specification for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for Internet access, including e-mail, the World Wide Web, news groups, and Internet Relay Chat (IRC). While Internet access has been possible in the past, different manufacturers have used different technologies. In the future, devices and service systems that use WAP will be able to interoperate.

Wireless Markup Language (WML) Formerly called Handheld Devices Markup Languages (HDML), this is a language that allows the text portions of Web pages to be presented on cellular telephones and personal digital assistants (PDAs) via wireless access. WML is part of the Wireless Application Protocol (WAP) that is being proposed by several vendors to standards bodies.

WSP Web services provisioning.

X.509 A widely used specification for digital certificates that has been a recommendation of the ITU since 1988.

XACML (Extensible Access Control Markup Language) A standard in encoded data exchange, makes possible a simple, flexible way to express and enforce access control policies in a variety of environments, using a single language.

XKMS (XML Key Management Specification) Leverages the Web Services framework to make it easier for developers to secure inter-application communication using public key infrastructure (PKI). XML Key Management Specification is a protocol developed by W3C that describes the distribution and registration of public keys. Services can access an XKMS-compliant server in order to receive updated key information for encryption and authentication.

XML (Extensible Markup Language) A flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. For example, computer makers might agree on a standard or common way to describe the information about a computer product (processor speed, memory size, and so forth) and then describe the product information format with XML. Such a standard way of describing data would enable a user to send an intelligent agent (a program) to each computer maker's Web site, gather data, and then make a valid comparison. XML can be used by any individual or group of individuals or companies that want to share information in a consistent way.

XML Encryption A process for encrypting and decrypting parts of XML documents. Most of today's encryption schemes use transport-level techniques that encrypt an entire request and response stream between a sender and receiver, offering zero visibility into contents of the interchange to intermediaries. Content-level encryption converts document fragments into illegible cipher text, while other elements remain legible as plain text.

XMLDSIG (XML Digital Signature) A W3C recommendation that defines an XML syntax for digital signatures. Functionally, it has much in common with PKCS#7 but is more extensible and geared towards signing XML documents. It is used by various Web technologies such as SOAP, SAML, and others.

XrML (Extensible rights Markup Language) A machine-interpretable language, developed at Xerox PARC. It uses XML for its syntax and was previously known as DPRL. XrML is intended to be a general purpose rights language to create usage licenses or specify the rights for a digital item. XrML is a core component in enabling distribution of digital content and access to digital services such as in an e-commerce context.

XSL (Extensible Stylesheet Language) A language for creating a style sheet that describes how data sent over the Web using the eXtensible Markup Language (XML) is to be presented to the user.

XSLT (Extensible Stylesheet Language Transformations) A language used to transform XML documents into other documents. In Second Site, XSLT is used to transform XML documents into HTML tags. The XSLT standard is administered by the World Wide Web Consortium (W3C).

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redbook.

IBM Federated Identity Manager manuals

- ▶ *IBM Tivoli Federated Identity Manager Release Notes Version 6.0*, GC32-1669-00
- ▶ *IBM Tivoli Federated Identity Manager Administration Guide Version 6.0*, GC32-1668-00
- ▶ *IBM Tivoli Identity Manager Federated Installation Guide Version 6.0*, GC32-1667-00

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 472. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014
- ▶ *Identity Management Design Guide with IBM Tivoli Identity Manager*, SG24-6996
- ▶ *Federated Identity Management and Secure Web Services*, REDP-3678
- ▶ *On Demand Operating Environment: An Overview and Implementation Guide*, REDP-3858
- ▶ *Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6*, SG24-6494

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Federated Identity Management white paper* (Heather Hinton, et al)
- ▶ *IBM Systems Journal on End-to-End Security*, Vol. 40, No. 31

- ▶ *IBM Tivoli Access Manager Base Administration Guide Version 5.1, SC32-1360*
- ▶ *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 5.1, SC32-1359*
- ▶ *IBM Tivoli Access Manager for e-business Plug-in for Web Servers Integration Guide Version 5.1, SC32-1365*
- ▶ *IBM Tivoli Access Manager for e-business Administration C API Developer Reference Version 5.1, SC32-1357*
- ▶ *IBM Tivoli Access Manager Administration Java Classes Developer Reference Version 5.1, SC32-1356*
- ▶ *IBM Tivoli Access Manager for e-business BEA WebLogic Server Integration Guide Version 5.1, SC32-1366*
- ▶ *IBM Tivoli Access Manager for e-business IBM WebSphere Application Server Integration Guide Version 5.1, SC32-1368*
- ▶ *IBM Tivoli Access Manager for e-business Problem Determination Guide Version 5.1, SC32-1352*
- ▶ *IBM Tivoli Access Manager for e-business Performance Tuning Guide Version 5.1, SC32-1351*
- ▶ *IBM Tivoli Access Manager for e-business IBM Tivoli Identity Manager Provisioning Fast Start Guide Version 5.1, SC32-1364*
- ▶ *IBM Tivoli Directory Server Installation and Configuration Guide Version 5.2, SC32-1338*
- ▶ *IBM Tivoli Directory Server Administration Guide Version 5.2, SC32-1339*
- ▶ *IBM Tivoli Directory Integrator 5.2: Reference Guide, SC32-1377*
- ▶ *IBM Tivoli Directory Integrator 5.2: Administrator Guide, SC32-1379*
- ▶ *Tivoli Identity Manager Policy and Organization Administration Guide Version 4.5, SC32-1149*
- ▶ *IBM Tivoli Identity Manager Tivoli Access Manager Agent for Windows Installation Guide Version 4.5, SC32-1165*

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ Web Services Security (WS-Security) specification
<http://www-128.ibm.com/developerworks/webservices/library/ws-secure/>
- ▶ Web Services Trust Language (WS-Trust) specification

- <ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf/>
- ▶ Web Services Provisioning (WS-Provisioning) specification
<http://www-128.ibm.com/developerworks/webservices/library/ws-provis/>
 - ▶ Web Services Federation Language (WS-Federation) specification
<http://www-128.ibm.com/developerworks/webservices/library/ws-fed/>
 - ▶ WS-Federation: Active Requestor Profile (WS-FEDACT) specification
<http://www-128.ibm.com/developerworks/webservices/library/ws-fedact/>
 - ▶ WS-Federation: Passive Requestor Profile (WS-FEDPASS) specification
<http://www-128.ibm.com/developerworks/webservices/library/ws-fedpass/>
 - ▶ Security in a Web Services World: A Proposed Architecture and Roadmap
<http://www-128.ibm.com/developerworks/webservices/library/ws-secmap/>
 - ▶ Security Assertion Markup Language (SAML), an OASIS standard
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
 - ▶ Liberty Alliance & WS-Federation: A Comparative Overview
<http://www.projectliberty.org/resources/whitepapers/wsfed-liberty-overview-10-13-03.pdf>
 - ▶ Multipurpose Internet Mail Extensions (MIME), RFC 2045
<http://www.faqs.org/rfcs/rfc2045.html>
 - ▶ Simple Authentication and Security Layer (SASL), RFC 2222
<http://www.faqs.org/rfcs/rfc2222.html>
 - ▶ The Simple and Protected GSS-API Negotiation Mechanism (SPNEGO), RFC 2478
<http://www.faqs.org/rfcs/rfc2478.html>
 - ▶ Generic Security Service API Version 2: Java Bindings, RFC 2853
<http://www.faqs.org/rfcs/rfc2853.html>
 - ▶ EXtensible Stylesheet Language (XSL) Tutorial
<http://www.w3schools.com/xsl/default.asp>
 - ▶ Security Assertion Markup Language (SAML) v1.1
<http://www.oasis-open.org/specs/index.php#samlv1.0>
 - ▶ Web Services Security 2, Username Token Profile 1.0 3, OASIS Standard 2004
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

- ▶ XML Stylesheet
<http://xml.apache.org/xalan-j/extensions.html#format-date-stylesheet>
- ▶ Command line utility
<http://xml.apache.org/xalan-j/commandline.html>
- ▶ WebSphere Application Server Version 6.0 Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

Access
 Control Lists 455
 Enforcement Point 455
 enforcement point 97
 Manager 173, 277, 283
 Rights 63
account
 creation 38
 de-linking 65, 108
 inking 107
 linking 60
 provisioning 42
accreditation process 16
ACL 455
ACS 206
active client 40
Address confirmation 16
AEF 97, 455
Agent 455
Anti-Virus Management 456
API 455
Application logout 167
Application Programming Interface 455
ARS 206
Assertion Consumer Service 206
Assertion Resolution Service 206
Association 455
Assurance 455
Asymmetric Keys 455
Attribute retrieval 109
Attribute Service 455
Audit 458
Authentication 456
Authentication credentials 45
Authentication services 89–90
Authorization 75, 456
Active Requestors 455
AZN API configuration 335

B

B2B 13, 153, 456
B2C 19, 22, 108, 456

B2E 19, 456
Base pattern 139
BASE64 306
benefits provider 33
BigCorp 194, 202, 219, 234, 293
BigCorpOne 33, 184
blacklist 191, 301
Browser Artifact 148
Browser Artifact Profile 149
Browser POST Profile 150
Business to Business 456
Business to Consumer 456
Business to Employee 456

C

CA 456
CDC 456
CD-SSO 105
Certificate 456
Certificate authority 456
CGI 200
Claim 457
Claim Confirmation 457
Common Domain Cookies 456
Common Object Request Broker Architecture 457
Container 457
CORBA 457, 460
Corporate Governance 14
Corporate identities 21
Corporate IT accounts 17
corporate view 21
Credential Exchange 74, 462
Credentials 64, 457
Credentials clean up 64
CUID 457
customer-for-life 178

D

DAML 176
Defederation 247
Demilitarized Zone 457
Desktop Single-Sign-On 194
dialtones 178

Digital Certificate 458
Directory 458
Directory Services 458
DMZ 457
Domain or Realm 458
DSMLv2 176

E

EAI

Authentication 370
Authentication Service 372
Enabled junctions 371
Example 371
Example setup 371
Headers 370–371
PAC 371
service 370
e-business 17
EJB 458, 460
End-to-end user lifecycle management 18
end-user 48
Enrollment costs 24
Enterprise JavaBeans 458
Enterprise Service Bus 26, 458
Enterprise Single Sign-On 136
ESB 26, 458
Extensible Markup Language 467
Extensible Stylesheet Language 467
External Authentication Interface 140

F

Federated Identity 4, 18
Federated Identity Business Models 9
Federated Identity Management
see FIM
Federated Single Sign-On 27, 136, 291
Federated SSO 137
Federation 458
standards and interoperability 58
user care 38
Federation Termination Notification 54
FIM 4, 33, 51, 81, 458
definition phase 81
Firewall 458
F-SSO 27, 105
FTN 54, 458

G

Generic Security Services Application Program Interface 459
Global goodbye 64
GSS-API 201, 459

H

HTTP Point of Contact 459

I

IBM

IBM Tivoli Access Manager 465
IBM Tivoli Access Manager for e-business 172
IBM Tivoli Directory Integrator 176
IBM Tivoli Directory Server 465
IBM Tivoli Federated Identity Manager
see TFIM
IBM Tivoli Identity Manage 174
IBM Tivoli Identity Manager 465
IBM WebSphere Application Server 138
offerings 85
Identity 459
assertion 42
Authentication Credentials 45
Business to consumer identities 22
common unique identifier 61, 71
Corporate E-mail 22
Corporate identities 21
CRM accounts 22
CUID 61, 71
Desktop identities 21
end-to-end identity 21
HR accounts 22
IPI 54
Legacy accounts 22
management 17, 459
Management costs 4
multiple identity account 23
Network identities 21
Oracle accounts 22
Password management costs 25
PIN number 25
Portal accounts 22
Profile Attributes 45
provider functionality 110
Provider Integration 172, 175–177
Provider Introduction 54, 459
Provider Specific Attributes 45

- Supply Chain 22
- Transaction Attributes 45
- verification 16
- WebSphere accounts 22
- Identity Federation 3, 75
 - Architecting 31
- Identity management system 98
- Identity mapping 459
- Identity Provider
 - see Idp
- IdP 459
- IE 459
- Implementation flow 80
- IMS 98
- Integrated Solutions Console
 - see ISC
- Internet 51
- Internet Explorer 199, 459
- Internet Information Server 199
- Intranet 198
- Intrusion Defense 459
- IPI 459
- ISC 179, 459

J

- J2EE 147, 460
- JAAS 307
- Java 2 Platform Enterprise Edition 460
- Java Database Connectivity 460
- Java JAR 206
- Java Naming and Directory Interface 460
- Java Security 460
- Java Server Page 460
- JDBC 460
- JITP 193
- JKS key stores 320
- JNDI 460
- JSP 296, 460
- just-in-time-provisioning 193

K

- Kerberos 201, 460
- Key
 - Escrow 460
 - Management 74, 460
 - Recovery 461
 - Services 96
- Key management 166

L

- LDAP 89, 460–461
- LECP 461
- Liberty 29, 54, 151, 187, 245, 461
 - ID-FF 59
 - Liberty enabled Client/Proxy 461
 - Liberty Federation Termination Identification 458
 - Liberty ID-FF 52
 - Liberty single sign-on 245
 - Liberty single sign-out 465
- Lightweight
 - pattern 143, 145
 - SSO 138
 - Third Party Authentication 461
- Lightweight Third Party Authentication 461
- Log-out 63
- LTPA 461

M

- Mapping Rules 461
- MASS 79, 461
- Method for Architecting Secure Solution 79
- middleware 21
- Mobile Station International ISDN Number 461
- MSISDN 461

N

- Name de-federation 108
- Network Security Solutions 461
- Non-repudiation 461

O

- OASIS 51, 57–58
- ODBC 460
- On 462
- On demand 462
 - Identity Management 74
 - integration 19
 - interoperation 19
 - ODOE 462
 - Operating environment 462
 - Policy Management 73
 - security infrastructure 73
- Open Platform for Security Check Point 462
- OPSEC 462
- Outsourced provider services 12

P

Pain point

- Compliance exposure 23
- Improve confidence 23
- Increase confidence 23
- Lower administrative cost 23
- Poor Market Reach 23
- Risk exposure 23
- Security exposure 23

Partnership 12

Partnership-based solutions 19

Passive Requestor 462

password management 38, 42

Password synchronization 108

PEP 462

PKI 462–463

Plug-in pattern 142

PoC 459

Point of Contact 138, 462

Point-to-point pattern 154

Policy Enforcement Point 172, 462

Policy Management 462

Portal 198

Portal-based integration 13

Privacy Policies 462

Profile 462

Profile Attributes 47

Project

- Build 81
- Definition 81
- Design 81
- Initiation 80

Provisioning 23

- Access rights 24
- authoritative source 47
- Create account 24
- Credentials 24
- De-provisioning 24
- Enrollment of user 24
- Initial password/PIN 24
- management costs 25
- Password synchronization 24
- Provisioning authentication credentials 47
- Registration of user 24
- user ownership costs 23
- User provisioning 23

Pseudonym Service 463

Public Key 463

Public Key Infrastructure 463

Pull protocol 105

Push protocol 105

R

RA 463

RBBanking 185, 245, 288

RBBenefits 33, 184–185

RBStocks 293, 302

RBTelco 219, 245, 287, 293, 302

RBTickets 185, 245, 288

RBTravel 194, 202

Realm or Domain 463

Redbooks Web site 472

- Contact us xix

Register Name Identifier 54

Registration Authority 463

RFC 2478 201

RNI 54

Role-Based Access Control 463

Roles

- Identity Provider 41
- Service Provider 41

Router 463

S

SAML 29, 51–52, 58–59, 291, 464

- assertion 190, 221, 294
- assertions 52
- bindings and profiles 52
- JITP 186, 193
- security token 302
- token 201

SASL 465

Scenario 184

Secure Conversation 456

Secure Logging 463

Secure Networks and Operating Systems 463

Secure Sockets Layer

- see SSL

Security

- Policy Expression 464
- STS 464
- Token 464
- Token Service 464
- triangle 20

Security Assertion Markup Language

- see SAML

Security Token Service 152

- Self-assertion 16
- Service
 - End Point Policy 464
 - oriented architecture 28
 - Provider 465
 - provider automation 13
 - Provider Integration 173, 175–176, 178
- Service Oriented Architecture 6, 464
- Services
 - Authorization Services 97
 - based delivery model 28
 - Identity Services 97
 - Key Services 96
 - Session Management Services 97
 - single sign-on services 97
 - Trust Services 92
- Session Management 63
- Session timeout 166
- Security Token Service 464
- Sharing authentication credentials 43
- Sharing profile attributes 43
- Sharing transactional attributes 43
- Shibboleth 51, 53
- Signature 464
- Signed Security Token 464
- sign-in 464
- sign-out 464
- Simple and Protected GSS-API Negotiation Mechanism 464
- Simple Authentication and Security Layer 465
- Simple Object Access Protocol
 - see SOAP
- single log-out 54
- single sign-off 55, 106
- single sign-on 227
- Single-Sign-On 193
- SLO 54, 465
- Smart card 465
- SMS 89
- SOA 6, 464
- SOAP 141, 195, 465
 - message exchanges 57
 - message security 57
- SOAP/HTTP 247, 262
- SP 465
- SPI 465
- SPNEGO 194, 296, 464
- SPS 465
- SSL 50, 52, 141, 195, 464
 - end-to-end 52
 - granularity 52
- SSO 54–55, 105, 193, 465
 - protocol functionality 105
 - protocol service 465
- Stateful Packet Inspection 465
- Switch 465

T

- T IM 465
- TAM 465
- TAMTokenGenerator 323
- TCPMON 357, 359
- TDS 465
- TFIM 465
 - Federated User Lifecycle Management 37, 86
 - Web Services Provisioning Management 37, 86
 - Web Services Security Management 37, 86
- TIM 174
- Tivoli
 - Tivoli Access Manager Family 171
 - Tivoli Directory Integrator 171
 - Tivoli Directory Server 171
 - Tivoli Identity Manager 171
 - Tivoli Access Manager binary security token 302
 - Tivoli Access Manager for eBusiness 136
- TLS 465
- Transactional attributes 47
- Transport Layer Security 52, 465
- Trust 465
 - Circle of Trust 457
 - Cryptographic elements 92
 - foundation of trust 15
 - Partner accreditation 15
 - Partner identity proofing 15
 - Partner reputation evaluation 15
 - Security token 92
 - Trust and Assurance 15
 - Trust Domain 465
 - trust infrastructure 39
 - Trust Modeling 465
 - trust relationships 19
 - Trust Service 51
 - Trust Services 92
 - Trusted Third Party 466
- TSL 52

U

UDDI 319
Uniform Resource Identifier 466
Uniform Resource Locator 466
URI 466
URL 466
user account creation 42
user enrollment 38
User lifecycle management of identities 18
User provisioning 23
User registration 24
Username token 302

V

Validation Service 466
Virtual Organization Polices 466

W

WAP 466
WAYF 109, 466
Web 466
Web Services 466
 Description Language 466
 Provisioning 467
 Security Management 190, 291
 Security roadmap 28
 Security Specifications 28
 Trust 466
 WSDL 466
 WS-Federation 28
 WS-Policy 28
 WS-Security 28, 291, 466
 WS-Trust 28, 292, 466
Web services provider 156
Web services requestor 155
WebSEAL 89, 172, 203, 223, 296
WebSphere 291
WebSphere Application Server 178, 185
Where are you from 109
Wireless Application Protocol 466
 WAP 466
Wireless Markup Language 467
 WML 467
WS Federation
 WS-FED 55
 WS-FEDACT 56
 WS-FEDPASS 56
WS-Federation 29, 52, 55, 59, 151, 186, 219, 227,

293

 Active 55
 Passive 55
WSP 467
WS-Security 57
WSSM 190
 Token Consumer 305
 Token Generator 304

X

X.500 461
X.509 142, 467
X509 Credential token 302
XML 467
XML signature 312
XSL 146, 160, 467



Redbooks

Federated Identity Management and Web Services Security

with IBM Tivoli Security Solutions



Redbooks

Federated Identity Management and Web Services Security

with IBM Tivoli Security Solutions

Introduction to Web services security standards

Complete product architecture and component discussion

Extensive federation business scenario

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 takes a close look 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.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6394-01

ISBN 0738492892