

Understanding SOA Security

Design and Implementation

Introducing an SOA security reference
architecture

Implementing scenarios based
on the IBM SOA Foundation

Deploying SOA using IBM
Tivoli security solutions



Axel Buecker
Paul Ashley
Julien Bouyssou
Gianluca Gargaro
Sridhar Muppidi
Ray Neucom
Neil Readshaw
Gregor Schinke



International Technical Support Organization

**Understanding SOA Security
Design and Implementation**

February 2007

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

First Edition (February 2007)

This edition applies to Version 6.0 of IBM Tivoli Access Manager for e-business, Version 6.1 of IBM Tivoli Federated Identity Manager, and Version 6.0 of IBM Tivoli Directory Server. We are also discussing several other IBM software products in the context of hands-on scenarios.

© Copyright International Business Machines Corporation 2007. 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	ix
Trademarks	x
Preface	xi
The team that wrote this IBM Redbook	xi
Become a published author	xiv
Comments welcome	xiv
Part 1. Business context and foundation	1
Chapter 1. Business context	3
1.1 Service orientation	4
1.1.1 More than componentization	5
1.1.2 Evolution not revolution	7
1.1.3 A focus on reuse	8
1.2 Security considerations for SOA	8
1.2.1 User and service identities and propagation	9
1.2.2 Connect to other organizations on a real-time, transactional basis ..	9
1.2.3 Composite applications	10
1.2.4 Managing identity and security across diverse environments	10
1.2.5 Protecting data	11
1.2.6 Demonstrable compliance with a growing set of standards	11
1.3 Security in the service-oriented life cycle	12
1.3.1 Security encompasses all aspects of the life cycle	14
1.4 Summary	15
Chapter 2. Architecture and technology foundation	17
2.1 Service-oriented architecture overview	18
2.1.1 Definition of a service-oriented architecture	18
2.1.2 Basic components of an SOA	19
2.2 IBM SOA Reference Model	21
2.3 The need for security in the SOA	24
2.4 IBM SOA Security Reference Model	28
2.4.1 IT Security Services	30
2.4.2 Security Policy Infrastructure	38
2.4.3 Business Security Services	43
2.5 IBM SOA Security Logical Architecture	49
2.5.1 Foundation scenarios	49
2.5.2 Typical deployment architecture	53

2.5.3 IBM SOA Security Logical Architecture summary	55
2.6 Conclusion	57
Part 2. IBM SOA Foundation scenarios	59
Chapter 3. IBM SOA Foundation Service Creation scenario	61
3.1 Scenario overview	62
3.1.1 Direct exposure architectural pattern	62
3.1.2 Indirect exposure architectural pattern	65
3.2 Applying the IBM SOA Security Reference Model	66
3.2.1 IT Security Services	67
3.2.2 Security Policy Infrastructure	78
3.2.3 Business Security Services	82
3.3 Summary	88
Chapter 4. IBM SOA Foundation Service Connectivity scenario	89
4.1 Scenario overview	90
4.2 Applying the IBM SOA Security Reference Model	94
4.2.1 IT Security Services	95
4.2.2 Security Policy Infrastructure	108
4.2.3 Business Security Services	111
4.3 Summary	116
Chapter 5. IBM SOA Foundation Service Aggregation scenario	117
5.1 Scenario overview	118
5.1.1 Overview of the Service Aggregation scenario	118
5.1.2 Web single sign-on perspective	119
5.1.3 Web services perspective	121
5.2 Applying the IBM SOA Security Reference Model	122
5.2.1 IT Security Services	124
5.2.2 Security Policy Infrastructure	139
5.2.3 Business Security Services	143
5.3 Summary	147
Part 3. Securing the Service Creation scenario	149
Chapter 6. Business scenario	151
6.1 Business model	152
6.1.1 Overview	152
6.1.2 Initial context - ITSOTelco	153
6.1.3 Initial context - ITSOBank	154
6.1.4 Preliminary SOA engagement	155
6.1.5 Business logic	157
6.1.6 Authentication and authorization	158

6.2 Business requirements	158
6.3 Technical requirements	159
6.3.1 Security requirements	160
6.3.2 Other functional requirements	161
6.3.3 Other non-functional requirements	162
Chapter 7. Solution design	163
7.1 Solution architecture introduction	164
7.2 IT Security Services	166
7.2.1 Identity Services	167
7.2.2 Authentication and authorization services	170
7.2.3 Confidentiality and integrity services	180
7.2.4 Audit Services	183
7.3 Security Policy Infrastructure	186
7.3.1 Policy Administration	186
7.3.2 Policy Decision and Enforcement	187
7.3.3 Monitoring and reporting	188
7.4 Business Security Services	189
7.4.1 Governance, risk, and compliance	189
7.4.2 Trust Management	189
7.4.3 Identity and access	190
7.4.4 Data protection and disclosure control	190
7.4.5 Secure systems and networks	190
7.5 Conclusion	192
Chapter 8. Technical implementation	193
8.1 Implementation scope	194
8.2 Configure security for the ITSO Banking Application	196
8.2.1 Import the application into Rational Software Architect	196
8.2.2 Key stores	201
8.2.3 Configure the application client	202
8.2.4 Configure the application	224
8.2.5 Export the application with security configuration	244
8.3 Deploying the ITSO Banking Application	244
8.3.1 Installing the CICS ECI resource adapter	245
8.3.2 Configuring the CICS Connection Factory	248
8.3.3 Configure a JAAS login module	253
8.3.4 Deploy the ITSO Banking Web service	255
8.4 Configure Web Service Security Management	259
8.4.1 Configure the WSSM trust chains	259
8.5 Running the scenario	288
8.6 Common Auditing and Reporting Service configuration	307
8.6.1 Configure Federated Identity Manager central auditing	309

8.6.2 Configuring central auditing for trust service events	314
8.7 Conclusion	322
Appendix A. Introduction to service-oriented architecture	323
Service-oriented architecture overview	324
Definition of a service-oriented architecture	324
Challenges and drivers for SOA	326
Why SOA now	330
SOA approach for building a solution	333
Getting started with SOA	335
SOA adoption	335
IBM SOA entry points	336
IBM SOA Foundation	338
Web services and SOA	338
Web services technologies	338
Web services and SOA	343
Appendix B. IBM SOA Foundation	345
SOA Foundation overview	346
SOA Foundation life cycle	346
Model	347
Assemble	348
Deploy	348
Manage	349
Governance	349
SOA Foundation Reference Architecture	350
SOA Foundation scenarios	354
Service Creation scenario	357
Service Connectivity scenario	360
Interaction and Collaboration Services scenario	367
Business Process Management scenario	367
Information as a Service scenario	368
Appendix C. Security standards and technology	371
Web services security specifications	372
WS-Security	373
WS-Policy	375
WS-Trust	376
WS-Federation	376
WS-SecureConversation	377
WS-SecurityPolicy	378
WS-Provisioning	378
More information	379
Security Assertion Markup Language	379

Liberty	380
eXtensible Access Control Markup Language	380
Java Authorization Contract for Containers	381
Service Provisioning Markup Language	382
Identity Attribute Service (IdAS)	384
z/OS Security	384
System Authorization Facility	384
RACF	385
Appendix D. Additional material	389
Locating the Web material	389
Using the Web material	389
Installing the CICS portion of the ITSObank scenario	390
Related publications	397
IBM Redbooks	397
Other publications	398
Online resources	398
How to get IBM Redbooks	398
Help from IBM	398
Index	399

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

Trademarks

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

Component Business Model™	IMS™	RUP®
CICS Connection®	OMEGAMON®	SecureWay®
CICS®	OS/390®	System z™
DataPower®	Rational Unified Process®	Tivoli®
DB2®	Rational®	WebSphere®
developerWorks®	Redbooks™	Workplace™
Everyplace®	Redbooks (logo)  ™	z/OS®
IBM®	RACF®	

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates.

DataStage, are trademarks or registered trademarks of Ascential Software Corporation in the United States, other countries, or both.

SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, Java, J2EE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Microsoft, Visio, Windows, and the Windows logo are trademarks of Microsoft 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.

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

Preface

Securing access to information is important to any business. Security becomes even more critical for implementations structured according to service-oriented architecture (SOA) principles, due to loose coupling of services and applications, and their possible operations across trust boundaries. To enable a business so that its processes and applications are flexible, you must start by expecting changes in both to process and application logic, as well as to the policies associated with them. Merely securing the perimeter is not sufficient for a flexible on demand business.

In this IBM® Redbook security is factored into the SOA life cycle reflecting the fact that security is a business requirement, and not just a technology attribute. We discuss a SOA security model that captures the essence of security services and securing services. These approaches to SOA security are discussed in the context of some scenarios, and observed patterns. We also discuss a reference model to address the requirements, patterns of deployment, and usage, and an approach to an integrated security management for SOA.

This IBM Redbook is a valuable resource to senior security officers, architects, and security administrators.

The team that wrote this IBM Redbook

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

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 20 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.

Paul Ashley is a Consulting IT Specialist working in the SOA Advanced Technology Asia Pacific team, part of the IBM Software Group. The team specializes in new SOA engagements and technology. Paul has worked in IT for 16 years, and holds degrees in Electronics Engineering, Computer Science, and a PhD in Information Security. Before joining the SOA Advanced Technology

team in January 2006, Paul worked as a Consulting IT Specialist for IBM Tivoli® Security for six years in both the USA and Australia.

Julien Bouyssou is an IT Architect in the Tivoli Security Techworks team, in SouthWest EMEA and is based in Paris, France. Before joining Tivoli last year Julien worked as an IT Architect for IBM Global Services for five years. He holds a degree in Information Technology and his areas of expertise include IT Architecture, application and enterprise security.

Gianluca Gargaro is an IT Specialist working in the Security Products Support Team in Rome, Italy. He has eight years of experience in Web application, network, and enterprise security. He holds a degree in telecommunication engineering from the University of Rome. His areas of expertise include IBM Tivoli Access Manager, IBM Tivoli Identity Manager, and IBM Tivoli Federated Identity Manager for which he is also an IBM Certified Advanced Deployment Professional. He has been a speaker at several technical security conference in Europe, the USA, and at the Tivoli briefing center in Rome.

Sridhar Muppidi is a Senior Security Architect at IBM Software Group. He is responsible for SOA Security Architecture and SOA security solutions. He is a product architect for the SOA Security Policy Management solution. As a part of world-wide security architecture and solutions design group, his responsibilities also include providing secure and manageable e-business solutions to enterprises, which includes architecting solutions for customers, working on new product development, and standards work. He holds a Ph.D in computer science and has published extensively.

Ray Neucom is the Lead Product Manager for Federation and Access Solutions in IBM Tivoli. 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 Bachelor of Science, a Master of Scientific Studies (both in Computer Science), and a graduate Diploma in Business Administration.

Neil Readshaw is a Senior Security Architect in Tivoli's Worldwide Customer Solutions (SWAT) team. He is based in the Gold Coast, Australia. He has 14 years of experience in software development, network management, information security, and systems integration. He holds degrees in Computer Systems Engineering and Computer Science from the University of Queensland, as well as the Certified Information Systems Security Professional (CISSP) certification. He has written extensively for the Tivoli Developer Domain on the IBM developerWorks® site.

Gregor Schinke is an Advisory IT Specialist in Software Group Services for Tivoli. He is based in Zurich, Switzerland. He has five years of experience in software development, Web application security, and systems integration. He holds a degree in Computer Science.



Figure 1 From left: Sridhar, Gregor, Axel, Gianluca, Paul, Ray, Neil, and Julien

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

Robert Haimowitz, Chris Rayns, Richard Conway, David Bennin, Emma Jacobs
International Technical Support Organization

Nicholas G. Harlow, Eric Wood, Timothy Hahn, Leigh Compton, Richard Salz,
David Shute, Barry Mosakowski, Heather Hinton, Venkat Raghavan, Alexander
Amies, John Ganci, Jeffrey Miller, Avery Salmon

Members of the Software Group Architecture Board Working Group for SOA
Security, specifically, Anthony Nadalin, Nev Zunic, Rob High, Nataraj
Nagaratnam, Maryann Hondo, Tony Cowan, Ryan Fanzone, Phil Fritz, Charles
Carrington, Alex Montare and many others.

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 have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

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

- ▶ Use the online **Contact us** review IBM Redbook form found at:
ibm.com/redbooks
- ▶ Send your comments in an e-mail to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Part 1

Business context and foundation

In this part, we discuss the business context behind SOA Security and why executives should be concerned about every aspect of it.

In order to consistently produce security solutions for SOA environments, we introduce and discuss the IBM SOA Security Reference Architecture.



Business context

Today's business environment is undergoing dramatic change. Competitive pressure from traditional and non-traditional sources, the rapid emergence and growth of new channels, increasing pressure to outsource selected business processes, and demands for compliance with a plethora of new regulatory and legal requirements are all contributing to an ever growing demand for change.

Traditionally, many organizations have struggled to manage change. In order to survive and prosper in the coming years, these organizations will need to develop a capability to sustain a constant state of change and evolution. The ability of an organization's IT systems to cope with this level of change will be a significant factor in the organization's success in adapting to increasingly dynamic business environments.

1.1 Service orientation

Service orientation is increasingly being viewed as a means to better align business and IT objectives, and to better support the levels of flexibility and change required by the business. Following a service orientation, existing business processes are decomposed into discrete units of business function termed *services*. These services are then recombined into business processes in a more flexible manner. Such decomposition has led to the emergence of collaborative eco-systems¹, where the reconstructed processes often integrate services from partners, outsourced providers, and even customers.

Figure 1-1 illustrates the traditional approach for implementing business processes, and associated IT applications, with each organizational unit acting in isolation. Each business process has its own *proprietary* implementation of the business activities, which are often re-implemented in slightly different ways in other business processes and organizational units.

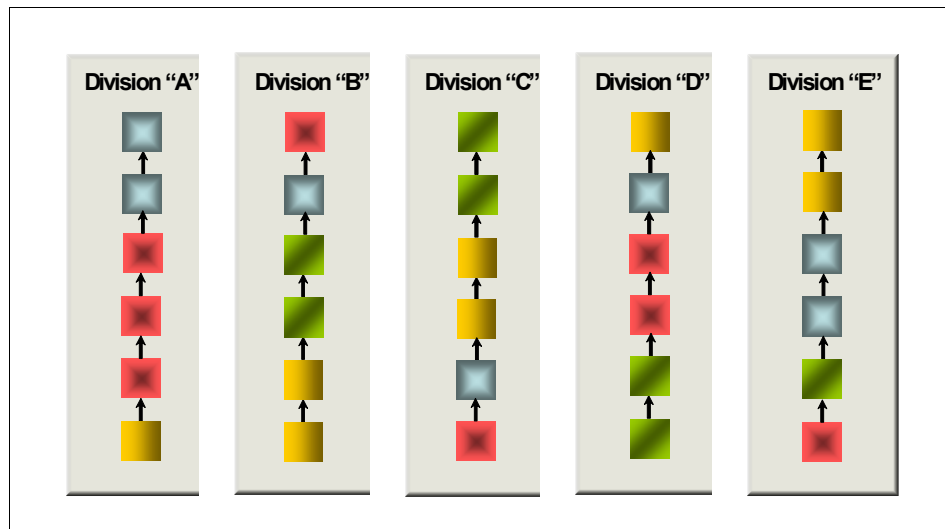


Figure 1-1 Traditional approach to business process design

Figure 1-2 on page 5 shows the goal of service orientation: common business logic is available in reusable services that can be performed where it is most appropriate, regardless of organizational boundaries.

¹ M. Lansiti and R. Levien, "The Keystone Strategy", HBS Press, Boston, MA (2004), "Daimler's New Way to Make Cars: Let Someone Else Do It", Forbes (August 16, 2004)

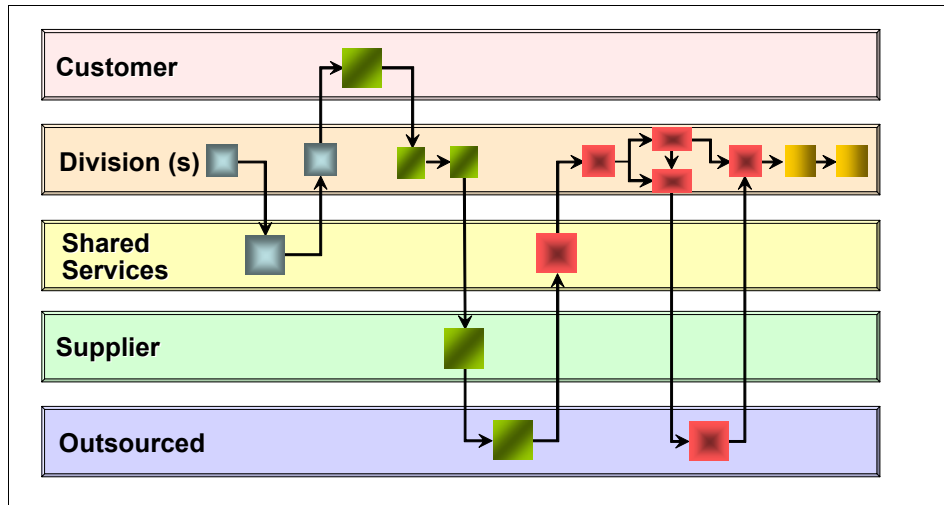


Figure 1-2 Service-oriented approach to business process design

1.1.1 More than componentization

Service orientation is more than merely decomposing business processes into components of business function. It also identifies that the focus in application development should be on implementing business logic, not on how components will interconnect. This allows services to be arbitrarily connected together to implement the desired business processes. Moreover, services can be replaced with equivalent services as required; for example, a particular business function may be outsourced, or there is a change in the partners used.

Consider the example shown in Figure 1-3, where each service needs to know how to connect to each other service that it may need to connect to. Given our goal of flexible connectivity, determining the scope of potential connectivity may be quite difficult to predict. This tight coupling between services makes the resulting business process fragile and difficult to change to meet the evolving needs of the business.

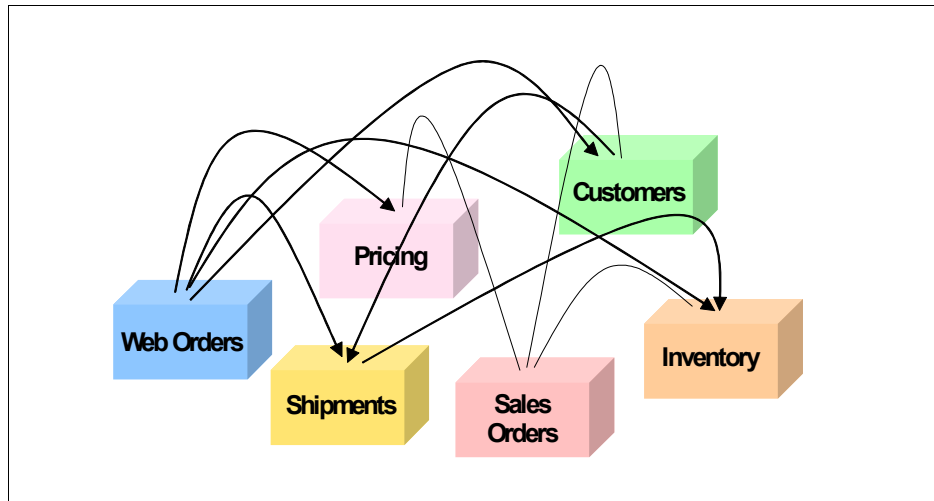


Figure 1-3 Directly connected services

By moving functionality such as flow control, translation of data formats and protocols, and identity propagation between services out of the application logic and into the services infrastructure, we gain greatly improved flexibility as to how services can be interconnected, as each service only needs to know how to connect to the service infrastructure, as shown in Figure 1-4.

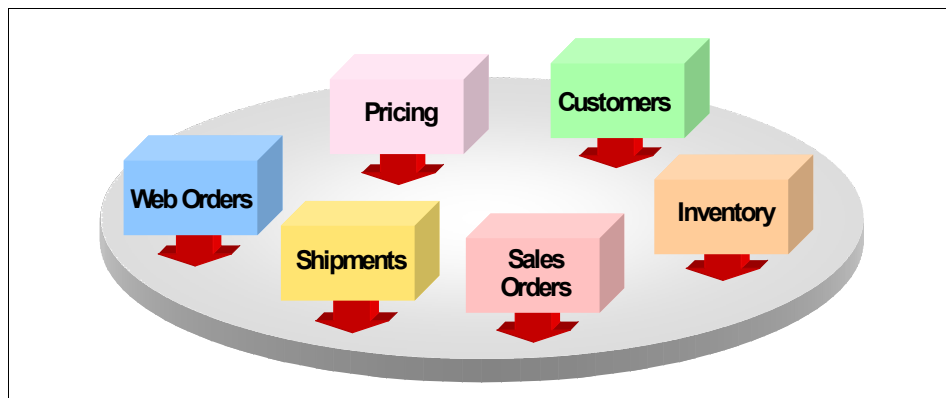


Figure 1-4 Connectivity using a service infrastructure

1.1.2 Evolution not revolution

Service orientation does not represent a radical change from previous approaches for designing distributed IT systems and attempting to align business and IT goals. Rather, it represents the next step in an evolution of message based approaches to connecting application components. Each progressive step has delivered greater degrees of flexibility and reuse by moving more of the non-business focused logic out of the application components.

As illustrated in Figure 1-5, the first step in this evolution was the introduction of a message queuing infrastructure, which abstracted connections between application components through the use of queues. Under this message queuing approach, instead of an application component talking directly with another component, it would connect via a message queue that can be directed to the target application component. Use of these queues removed the burden from the application developer to deal with the intricacies of different platforms and ensuring messages are reliably delivered when the target component may be temporarily unavailable or under heavy load. Message queuing approaches do not, however, address making sure that the information is passed in the correct format, nor do they facilitate routing of the messages based on the content of the information in the message.

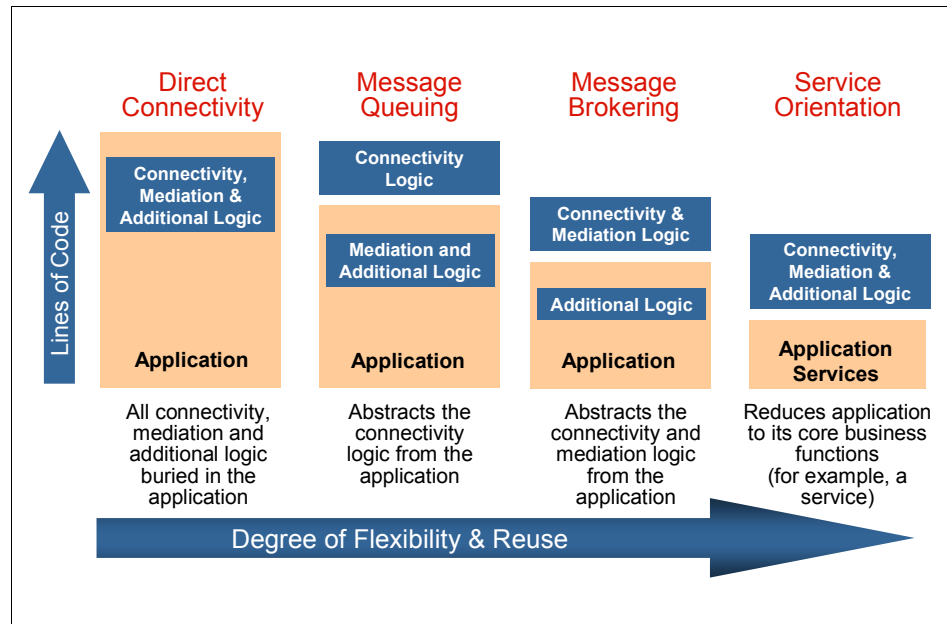


Figure 1-5 Service Orientation: the next step of connectivity evolution

The concept of a *broker* was introduced to add message transformation and routing functionality to the services provided by the message based infrastructure. These message brokers could also augment content and route messages based on the content of the messages. This resulted in more loosely connected application components, and less non-business logic needing to be developed for each component.

The development and broad acceptance of new open standards has allowed the next step in the evolution to become viable. With service orientation, the infrastructure services offered to application components are expanded to include locating and binding to services, more complete transparency of protocols and programming models, and more dynamic connectivity of services.

1.1.3 A focus on reuse

An important differentiation between service orientation and earlier distributed application development approaches is a much stronger focus on reusing existing IT assets. This emphasis on the value of reuse has made a transition to a service orientation more attractive and affordable for enterprises with large investments in existing IT systems, some dating back over 30 years.

Reusing existing systems does come at a price, however, the resulting systems comprise a heterogeneous mix of new and old technologies, which presents systems management, compliance, and security management challenges.

1.2 Security considerations for SOA

Examining the security management challenges, there are a number of considerations in a service-oriented environment:

- ▶ The need for user and service identities and propagation of these identities across the organization
- ▶ The need to seamlessly connect to other organizations on a real-time, transactional basis
- ▶ The need to ensure that, for composite applications, proper security controls are enacted for each service and when used in combination;
- ▶ The need to manage identity and security across a range of systems and services that are implemented in a diverse mix of new and old technologies
- ▶ Protection of business data in transit and at rest
- ▶ The need for demonstrable compliance with a growing set of corporate, industry, and regulatory standards

1.2.1 User and service identities and propagation

Service orientation aims to provide services that can be interconnected and reused as appropriate to fulfill a particular business process. Moreover, these services must be connected and implemented in a secure and auditable manner, according to a defined security policy. Identity and security therefore play key roles in delivering on the promise of service orientation.

Identities exist for both users and services, and both should be subject to the same security controls. Security policy defines the requirements for access to the provided services.

The identities may need to be propagated throughout the SOA environment. In many cases, service implementations may restrict the options and formats available for propagating a user's identity to/from the service. Identity services are therefore required in the infrastructure to deal with these identity mediation issues, so that services can be easily interconnected without worrying about how to map and propagate user identity from one service to the next.

Role based access control may also be utilized in some cases to dramatically reduce the user administration overheads and costs. In these cases, membership of users and services in a particular role will enable access, rather than being explicitly linked to the identity itself.

1.2.2 Connect to other organizations on a real-time, transactional basis

There are several forms of inter-organization interaction that may occur in a service-oriented deployment. A first step towards service orientation may involve integrating service user interfaces from different domains or organizations into a single portal interface. Another example may involve a service provided by a partner organization being directly invoked from a business process.

Regardless of the form of the interaction, it is imperative that security, identity, and access policies are defined and enforced for all transactions. These policies need to be enforced for both incoming and outgoing requests.

Secure boundary services are an obvious starting point. These boundary security services should be able to provide coarse grained verification that requests are coming from or going to trusted parties.

Establishing the trust relationship between the organizations is a key step in allowing inter-organization cooperation. This involves establishing rules around interaction, such as defining identity information that should be propagated

between organizations, as well as establishing the cryptographic keys used for the messages.

It is very unlikely that user identities will be the same in all of the service components in a business process flow that spans different organizations. Identity services will therefore be required to validate the identity of the requesting user, confirm that they are authorized to perform the requested operation, and map their identity to one that the target service can understand and use to identify the user or service making the request.

1.2.3 Composite applications

The security policy for the services include the rules established for allowing services to be accessed. A user or service may require specific privileges to allow them to access a service.

When services are combined, such as when they are choreographed into a higher level business process, the combination of these services may require a another examination of the security policy. For example, a user may be allowed to access Service A and Service B independently. However, when these two services are choreographed together, perhaps with other service invocations, then the user may no longer be allowed to access these services.

The complexity in an SOA environment is that the security policy for the choreographed services needs to take into account the mixing and matching of services in different combinations as required to reflect changes in business processes. Each new choreography may require examination of the security policy to ensure it remains valid for this new combination.

1.2.4 Managing identity and security across diverse environments

A typical service-oriented environment will have many points at which identity and security policy is enforced and implemented. These policy enforcement points will be located both at the service connectivity level as well as within the implementations of the services themselves. Management of a policy across these various heterogeneous enforcement points may be viewed as *swivel chair management* due to the need for an administrator to deal with a diverse set of management interfaces, and their associated security policy terminology and semantics.

For service orientation to achieve its goal of business flexibility within an environment of governance and regulatory compliance, definition and management of identity and security policy management should be simplified, with consistent management terminology and semantics across the various enforcement points being managed. This defined policy can then either be

directly enforced by the enforcement points or automatically translated into something that the endpoints can understand and enforce.

1.2.5 Protecting data

Protection of data from unauthorized modification and disclosure is a key requirement within an SOA. Data needs to be protected because it is business sensitive or privacy sensitive or both. A policy should therefore be in place to ensure that data is protected in both transit and at rest, with consistent security measures applied.

Data protection is especially important when data moves outside the organizational boundary, which may happen without the knowledge of the consumer. For example, an internal service may be replaced with an outsourced service, with data now flowing to the external organization. If the data is business or privacy sensitive then the service provider may need to ensure appropriate protection is in place to satisfy the policy requirements of the calling organization.

1.2.6 Demonstrable compliance with a growing set of standards

Auditing of transactions is required to provide the data needed for assessing compliance, which measures the performance of the IT environment relative to measures established by the business policies. This might include verifying the working system against a set of internally created policies, and also against external regulatory acts.

Complexity is increased in an SOA where different applications from different sources or vendors are targeted for different levels of compliance. This is especially true when accessing services provided by an external organization, where the regulatory and compliance regime is different from the requesting organization.

If possible, the audit data produced by the various policy enforcement points should be integrated into a single repository, or federated into a single logical view of the data. This will facilitate the production of the required audit reports, verification of compliance against policy, and investigation of security-related events.

1.3 Security in the service-oriented life cycle

An important facet of service orientation is an emphasis on the entire life cycle of IT systems—from conception to ongoing operation and management.

Service orientation aims to better align business and IT goals, and to provide greatly improved capability to deal with change. The service-oriented life cycle is therefore built around a business-driven development methodology that includes the following phases:

- ▶ **Model:** Use modeling tools to define the business process at a business function level, and model the actual services that will be part of an assembled, composite application.
- ▶ **Assemble:** Assemble the individual services and write the code that is needed to implement the business rules for the application. Pre-existing services can be re-used, new services can be developed, or both.
- ▶ **Deploy:** Deploy the services to runtime environments, such as transaction management engines like WebSphere® Application Server, CICS®, IMS™, and so forth. Use integration components, primarily an enterprise service bus (ESB), to link together the various services needed for the composite application. For more information about an ESB, refer to “Enterprise Service Bus” on page 353.
- ▶ **Manage:** Implement the management infrastructure for monitoring and managing the services and the service infrastructure. This includes not only IT management tools, but also business management and monitoring tools to measure actual business activities.
- ▶ **Governance** underpins all the life cycle stages, providing guidance and oversight for the entire service-oriented environment.

Figure 1-6 shows an overview of the service-oriented life cycle.

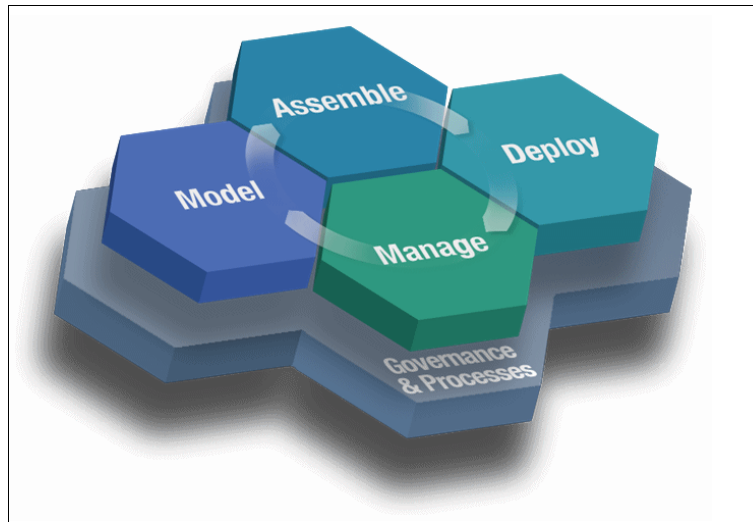


Figure 1-6 Service-oriented life cycle

1.3.1 Security encompasses all aspects of the life cycle

As shown in Figure 1-7, certain roles in an organization contribute towards the creation, definition, refinement, monitoring, verification, and management of security policies during the execution of the service-oriented life cycle.

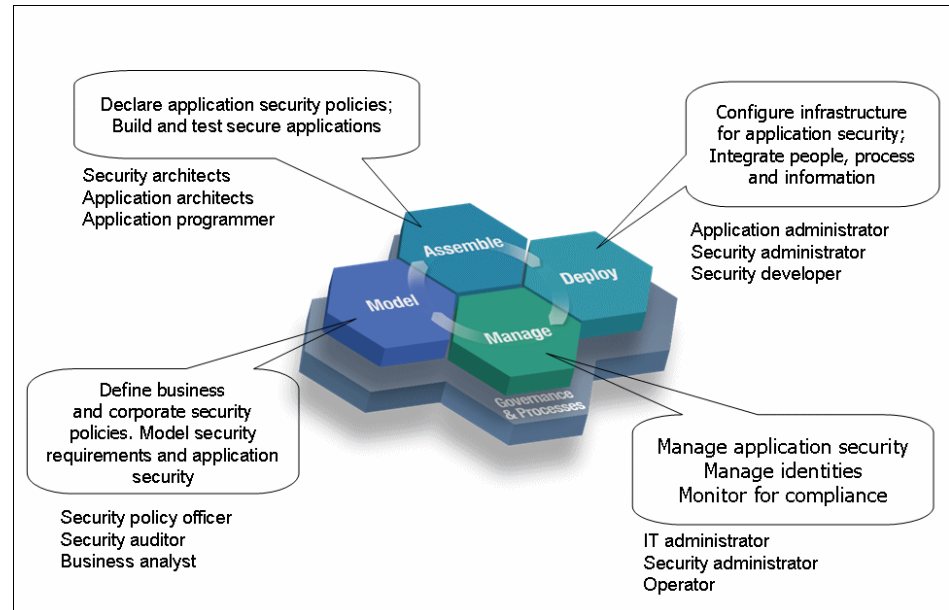


Figure 1-7 Service-oriented life cycle from a security perspective

Model

- ▶ Corporate security officers and equivalent executives defining corporate security policies and outlining regulations with which the business must comply.
- ▶ Business analysts working with security policy officers translating corporate security policies in terms of a business vocabulary and business process during the business process modeling phase, and providing a set of choices to be customized.

Assemble

- ▶ Application architects and security architects modeling the security, identity and access policies, based on choices provided by the business analyst.
- ▶ Application programmers and application administrators factoring in these security, identity, and access policies by declaring these requirements for the infrastructure to enforce, or when the infrastructure support is not sufficient, implementing them in their applications.

Deploy

- ▶ Application administrators installing the applications and working with security developers and security administrators to configure the applications and associated security policies.

Manage

- ▶ IT and security administrators managing the security policies across a set of applications and infrastructure to meet the requirements that may continue to change over time.
- ▶ Operators monitoring the system behavior for compliance, and detecting situations that are potential security threats and feeding that back to administrators to make changes as required.
- ▶ Business analysts viewing business dashboards to assess the impact to the business due to certain system security events.
- ▶ Security auditors assessing the system's compliance with regulatory and corporate policies.

It is significant to observe that security policies are specified and refined throughout the life cycle, undergoing transformations from one phase to the next.

1.4 Summary

Identity and security management permeate all aspects of the service-oriented life cycle and are key enablers for achieving the connectivity and flexibility goals of service orientation.

Examination of high-level requirements for identity and security management in a service-oriented environment identified the following aspects:

- ▶ The need for user and service identities and propagation of these identities across the organization
- ▶ The need to seamlessly connect to other organizations on a real-time, transactional basis
- ▶ The need to ensure for composite applications that proper security controls are enacted for each service and when used in combination
- ▶ The need to manage identity and security across a range of systems and services that are implemented in a diverse mix of new and old technologies
- ▶ Protection of data in transit and at rest
- ▶ The need for demonstrable compliance with a growing set of corporate, industry, and regulatory standards



Architecture and technology foundation

In this chapter, we describe how security can be applied to a service-oriented architecture (SOA). SOA applications are built from composable services across a distributed environment. Securing this environment is both critical and challenging.

We begin by introducing SOA and describe the *IBM SOA Reference Model*. The requirements for security in an SOA environment are outlined, showing security that spans the SOA. We then discuss the *IBM SOA Security Reference Model* and then apply this model to a typical deployment architecture to derive the *IBM SOA Security Logical Architecture*.

As a part of deriving the SOA Security Logical Architecture, we also introduce three of the SOA Foundation scenarios. These scenarios are used throughout the rest of the IBM Redbook to demonstrate how the IBM SOA Security Reference Model can be applied.

2.1 Service-oriented architecture overview

In this section we provide an overview for a service-oriented architecture (SOA) and introduce the important SOA terminology that is used throughout the rest of this IBM Redbook.

2.1.1 Definition of a service-oriented architecture

Figure 2-1 highlights the key terms used to describe a service-oriented architecture.

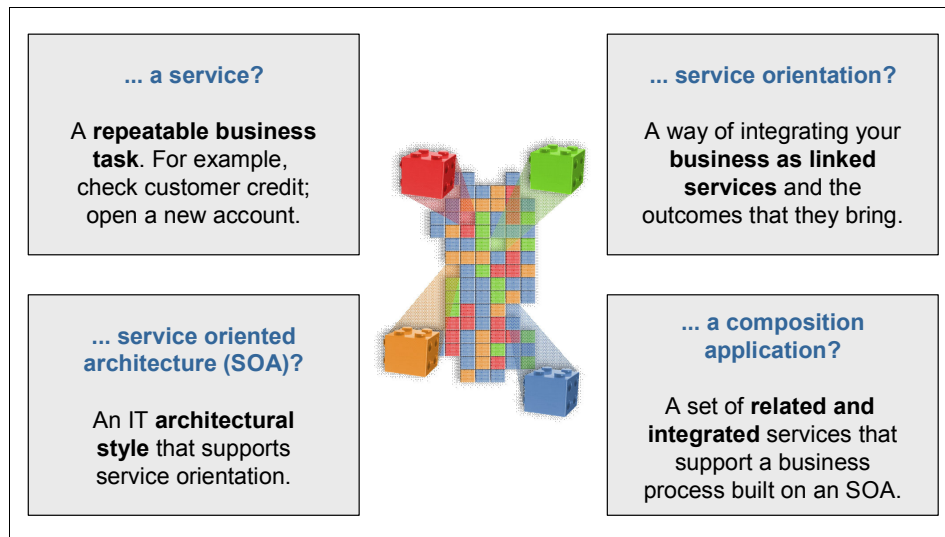


Figure 2-1 Definition of key terms for a service-oriented architecture

A *service* is representative of a repeatable business task. Services are used to encapsulate the functional units of an application by providing an interface that is well defined and implementation independent. Services can be invoked (consumed) by other services or applications.

Service Orientation defines a method of integrating business applications and processes as linked services.

Service-oriented architecture (SOA) can be different things to different people depending on the person's role and context (business, architecture, implementation, or operational). From a business perspective, SOA defines a set of business services composed to capture the business design that the enterprise wants to expose internally, as well as to its customers and partners. From an architectural perspective, SOA is an architectural style that supports

service orientation. At an implementation level, SOA is fulfilled using a standards based infrastructure, programming model, and technologies, such as Web services. From an operational perspective, SOA includes a set of agreements between service consumers and providers that specify the quality of service, as well as reporting on the key business and IT metrics.

A *composite application* is a set of related and integrated services that support a business process built on an SOA.

2.1.2 Basic components of an SOA

At the most basic level, an SOA consists of the following three components:

- ▶ Service provider
- ▶ Service consumer
- ▶ Service registry

Note: In Figure 2-2 and throughout the rest of this IBM Redbook, the terminology *service consumer* is used to denote the client that is calling the service. However, other deliverables and IBM Redbooks also use the term *service requestor*. Consider service consumer and service requestor to be the same.

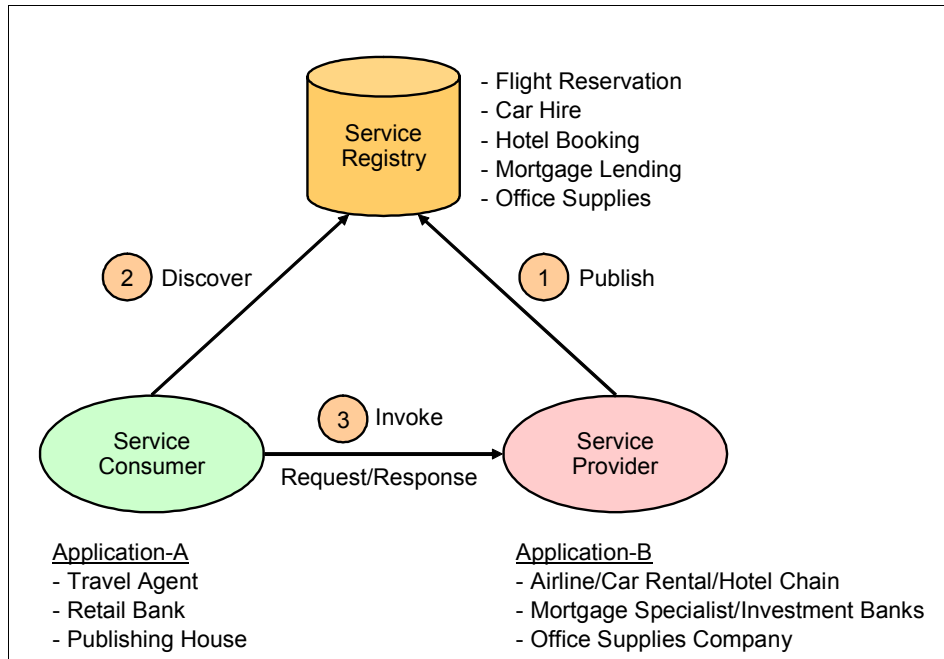


Figure 2-2 SOA components and operations

Each component can also act as one of the two other components. For example, if a service provider needs additional information that it can only acquire from another service, it acts as a service consumer. Figure 2-2 shows the operations each component can perform.

The service provider creates a service and in some cases publishes its interface and access information to a service registry.

Each provider must decide which services to expose, evaluate trade-offs between security and easy availability, determine how to price the services or figure out how to exploit the value of the services if they are free. The provider also has to decide under what category the service should be listed, and what sort of trading partner agreements are required to use the service.

The service registry is responsible for making the service interface and implementation access information available to service consumers.

The implementers of a service registry need to consider the implementation scope for the registry. For example, there are public service registries available over the Internet to an unrestricted audience, as well as private service registries that are only accessible to users within a company-wide intranet.

The service consumer locates (discovery) entries in the service registry and then binds to the service provider in order to invoke the defined service.

2.2 IBM SOA Reference Model

The IBM SOA Reference Model is shown in Figure 2-3. The core services of the architecture are:

- ▶ Interaction Services
- ▶ Process Services
- ▶ Information Services
- ▶ Business Application Services
- ▶ Access Services
- ▶ Partner Services

There are also support components of the core services. These are shown on the outside of the core services in the figure.

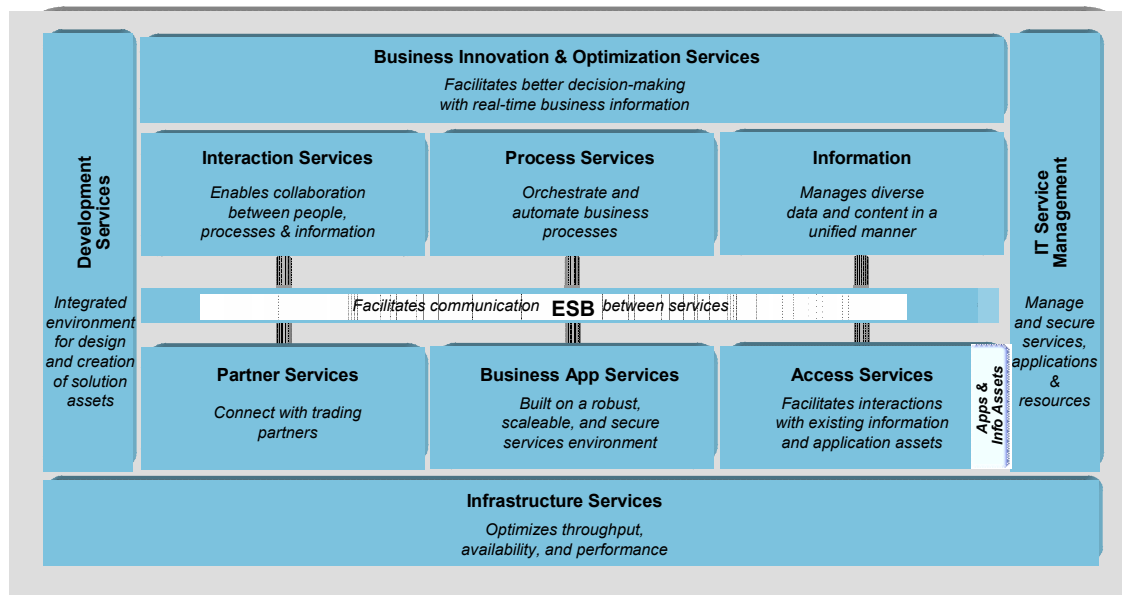


Figure 2-3 IBM SOA Reference Model

Interaction Services

Interaction Services provide the capabilities required to deliver IT functions and data to users, meeting their specific preferences.

Process Services

Process Services provide the control capabilities required to manage the flow and interactions of multiple services in ways that implement business processes.

Information Services

Information Services provide the capabilities necessary to federate, replicate, and transform disparate data sources.

Business Application Services

Business Application Services are called by service consumers. Service consumers include other components in the logical architecture, such as portal or business processes.

Access Services

Access Services provide bridging capabilities between core applications, prepackaged applications, enterprise data stores, and the Enterprise Service Bus (ESB) to incorporate services that are delivered through existing applications into an SOA.

Partner Services

Partner Services provide the document, protocol, and partner management capabilities for business processes that involve interactions with outside partners and suppliers.

Supporting components

This section includes a brief description of the support components:

- ▶ Enterprise Service Bus (ESB)
- ▶ Business Innovation and Optimization Services
- ▶ Development Services
- ▶ Infrastructure Services
- ▶ IT Service Management

Enterprise Service Bus

The Enterprise Service Bus (ESB) provides an infrastructure that removes the direct connection dependency between service consumers and providers. Consumers connect to the bus and not the provider that actually implements the service. This type of connection decouples the consumer from the provider. A bus also implements further value add capabilities, such as security and delivery assurance. It is preferable to implement these capabilities centrally within the bus at an infrastructure level rather than within the application.

At minimum, the ESB should have the following capabilities:

- ▶ **Routing:** Ensure that any request a consumer initiates is sent to the correct provider.
- ▶ **Addressing:** Addressing complements routing to provide location transparency and support service substitution. Service addresses are transparent to the service consumer.
- ▶ **Messaging Styles:** Should support a variety of messaging styles. The most common are request/response, fire and forget, events, publish/subscribe, and synchronous and asynchronous messaging.
- ▶ **Transport Protocols:** Support protocol transformation between consumer and provider.

Business Innovation and Optimization Services

Business Innovation and Optimization Services are primarily used to represent the tools and the metadata structures for encoding the business design, including the business policies and objectives.

Business Innovation and Optimization Services exist in the architecture to help capture, encode, analyze and iteratively refine the business design. The services also include tools to help simulate the business design. The results are used to predict the effect of the design, including the changes the design will have on the business.

Development Services

Development Services encompass the entire suite of architecture tools, development tools, visual composition tools, assembly tools, methodologies, debugging aids, instrumentation tools, asset repositories, discovery agents, and publishing mechanisms needed to construct an SOA based application.

Infrastructure Services

Infrastructure Services form the core of the information technology runtime environment used for hosting SOA applications. These services provide the ability to optimize throughput, availability, performance, and management.

IT Service Management

Once the application has been deployed to the runtime environment, it needs to be managed along with the IT infrastructure on which it is hosted. IT service management represents the set of management tools used to monitor service flows, the health of the underlying system, the utilization of resources, the identification of outages and bottlenecks, the attainment of service goals, the enforcement of administrative policies, recovery from failures, and the security of the system.

2.3 The need for security in the SOA

Security areas that have to be addressed in an SOA environment do not change from the ones already addressed when architecting other IT systems. One key difference though, is that because the IT systems are more tightly aligned with business processes in the SOA environment, security practices have to be adapted to be aligned to the business processes as well. Security policies have to face new challenges due to this alignment.

As discussed earlier in 1.3, “Security in the service-oriented life cycle” on page 12 and shown again in Figure 2-4, security must be factored from the beginning of an SOA, from model, assemble, deploy, and manage. This is a very important point and one aspect that is easily overlooked in a SOA.

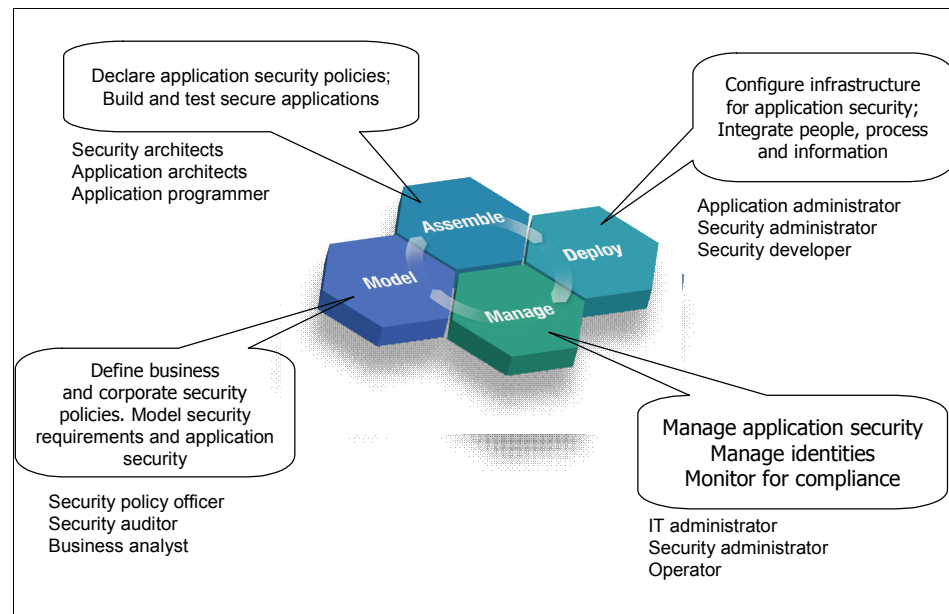


Figure 2-4 Applying security to the SOA development life cycle

Another view of applying security is shown in Figure 2-5 on page 25.

Beginning on the left side of the diagram, service consumers at the top of the diagram make requests to services. They may call these services directly, or alternatively they start a business process that calls services. A service's definition describes the service. A service may be an atomic service or made up of a composite set of services, where these composed services may in turn be atomic or composite services. Service components are used as intermediaries to expose existing application functions for those applications that cannot be

directly exposed. Whether directly or indirectly exposed, the application function itself is the target of the service consumer's request.

Now examining the right side of Figure 2-5, SOA security requirements should be included through all of the SOA implementation layers. Beginning at the service consumers, through the business processes, service definitions, service components, and finally service implementation, security needs to be considered.

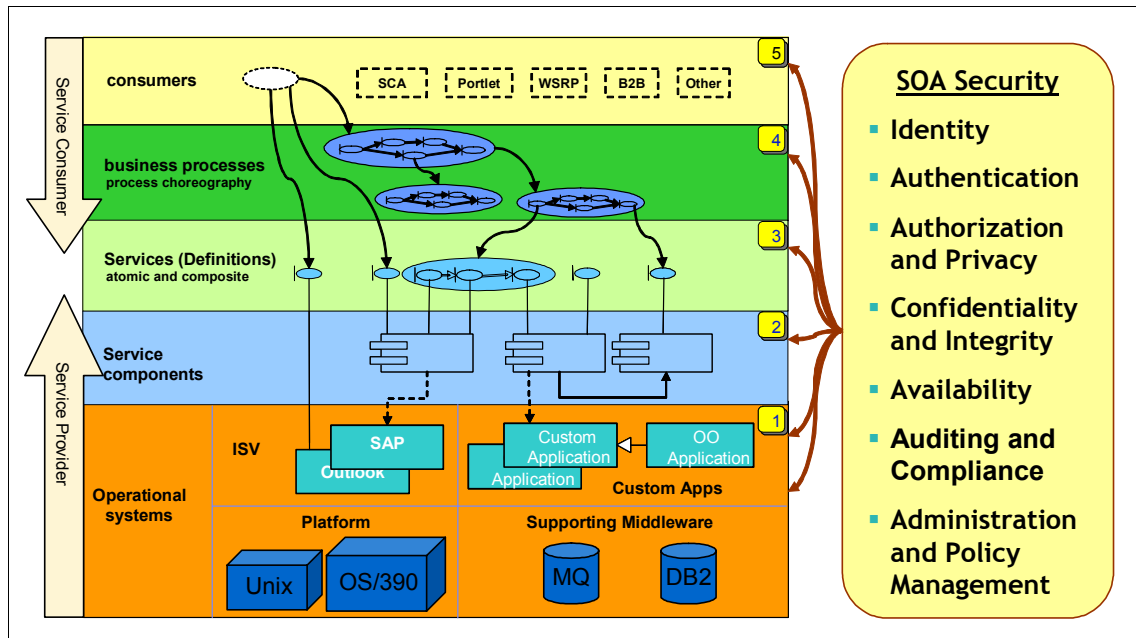


Figure 2-5 SOA security encompasses many aspects of security and should be applied from service consumer to service provider

Defining the terms in Figure 2-5:

► Identity

Generally, an identity is a property of an object that allows it to be distinguished from other objects. In our context, the object in question is a user or a consumer. The identity of these consumers is typically represented with a unique value, or identifier, such as a user name or a UUID. Within our SOA environment, there are many types of consumers, including suppliers, partners, internal staff, and applications. Each of these consumers needs to be correctly identified at all stages of an SOA environment.

The most obvious way for users (including suppliers, partners and internal staff) to identify themselves is by an explicit interaction, presenting an

identifier to a user interface, such as a Web portal. This information is then used to create an *identity token*, a unique representation of the identity and attributes of the consumer in a standardized format. When requests are issued by an application on behalf of a user, identity information is typically carried with the request in an identity token. The binding of an identity token to a request allows an identity to be carried with a request so that the service provider and intermediaries can identify the originator of the request.

As a consumer may not have the same identity at different points in a request flow, we need to be able to map identities contained within identity tokens. For example, a consumer may assert a user name (and password) to a portal. The consumer's identity token will represent this claimed identity. As part of fulfilling this request, invocation of a CICS-based resource may be required; this implies that the identity token needs to provide enough information to support the identification of this consumer in a RACF® ACEE format. Identity mapping techniques are employed to handle this type of identity mapping.

While many types of identity token are possible, common token types include Username Tokens, X.509 certificate based tokens, and tokens based on the Security Assertion Markup Language (SAML) assertion from the OASIS Security Service Technical Committee (<http://www.oasis-open.org>).

- ▶ **Authentication:** Assertion of an identity is often not sufficient to allow us to trust that the consumer really has the identity that they are claiming. The process of *authentication* is used to prove this claim. Authentication is the process of proving that the consumer legitimately has their claimed identity by evaluating additional information (*authentication credentials*) that is bound to this identity and can only be provided by a consumer with that identity. A typical example of this type of additional information is a password. This additional information can take the form of something that the consumer knows (for example, a password associated with a user name), something the user possesses (for example, a physical token capable of generating a one-time-use password or a certificate), or something that the user has (for example, biometric information, such as a fingerprint). The authentication process involves collecting the consumer's identity and authentication credentials and evaluating that the credentials presented by the consumer correspond to credentials that are expected to be presented by the user.

Not all forms of authentication require an interaction with a consumer. Authentication information may be bound to a request and carried with the request in the form of a *security token*. A security token differs from an identity token in that it includes additional information that allows it to be used as part of the authentication process. For example, a security token may contain a user name and password that can be evaluated using the same process that a user-presented user name and password are evaluated. These authentication credentials can be used to perform authentication of the consumer at components along a transaction path.

As a refinement of the security token, an identity token claims an identity without the additional authentication credentials. For example, consider a consumer who has already authenticated by direct interaction, by presenting a user name and password to a portal. Instead of including a security token (containing a user name and password), requests from the portal on behalf of the consumer may include an identity token only. Downstream components are expected to trust that the appropriate authentication has already taken place and do not require an additional authentication of the consumer.

- ▶ **Authorization and Privacy:** *Authorization* is the process of evaluating if an authenticated identity is allowed to have its request fulfilled. Authorization decisions may differ depending on the kind of user performing the access request. For example, the same service may be accessed by internal staff of the organization, but not by partners from different organizations. The internal staff member is allowed to access the service (they are authorized), while the partner is not allowed (they are not authorized). Authorization may need to be implemented at multiple points in the transaction path, as well as at different access layers such as gateways, application servers, applications, and databases.

Privacy is a special case of the authorization, where access to *Personally Identifiable Information* (PII) is controlled. This type of authorization is usually based on the data being retrieved. For example, it may be okay to access a customer's home address, but not their home phone number.

- ▶ **Confidentiality and Integrity:** Protection of data at rest, in transit, and in process are important considerations.

Confidentiality protection usually means encrypting the data so that it cannot be read by unauthorized persons. Only someone in possession of the decryption key can read the data. *Integrity* means protecting against data modifications that are undetected. This is usually achieved using digital signatures or message authentication codes. Note that signing of data in the form of a request or message provides a form of authentication, namely message authentication. The originator of the request will sign the message, both proving that they understand the contents of the request (based on their encrypting the message with their unique private signing key) and providing protection against modifications to the request (based on the inability to validate a signature against a modified message).

For example, the standard *WS-Security*, as discussed in Appendix C, "Security standards and technology" on page 371, can be used to protect Web services messages. It provides message confidentiality, integrity, and authentication.

Security data, such as passwords, encryption keys, configuration files, and application data stored in databases or similar places, may also be protected with confidentiality and integrity mechanisms.

- ▶ **Availability:** *Availability* of a service or resource implies that it is able to provide a response to a request in a timely manner. Ensuring that services are available when required is key in many SOA environments. Architecting for *high availability* includes application clustering, database clusters, and similar techniques.
- ▶ **Auditing and Compliance:** Ensuring that security related events are recorded, then analyzed and reported on is the domain of *auditing*. Often auditing is implemented in individual existing systems, so bringing that data together can be a key requirement. Checking that the security related events comply with the internal business policy is the domain of *compliance*. Compliance might also mean ensuring that the system follows legislative requirements.
- ▶ **Administration and Policy Management:** Rules about authentication, authorization, data protection, auditing, and so on are described in the security policies. As security policies may evolve to keep up with changing business policies, *administration and policy management* have to be flexible and easily configurable. Adding partners or a new set of services, applying dedicated and configurable security policies and deploying it easily, securely, and efficiently, must be made easier without having to reconsider the entire architecture.

2.4 IBM SOA Security Reference Model

The IBM SOA Reference Model is shown in Figure 2-3 on page 21. The *IT Service Management* supporting component applies to the entire reference model and its component includes security. The main components of security are the *Business Security Services*, *Security Policy Infrastructure*, and *IT Security Services*. These components together form the basis of the Reference Model, as shown in Figure 2-6 on page 29.

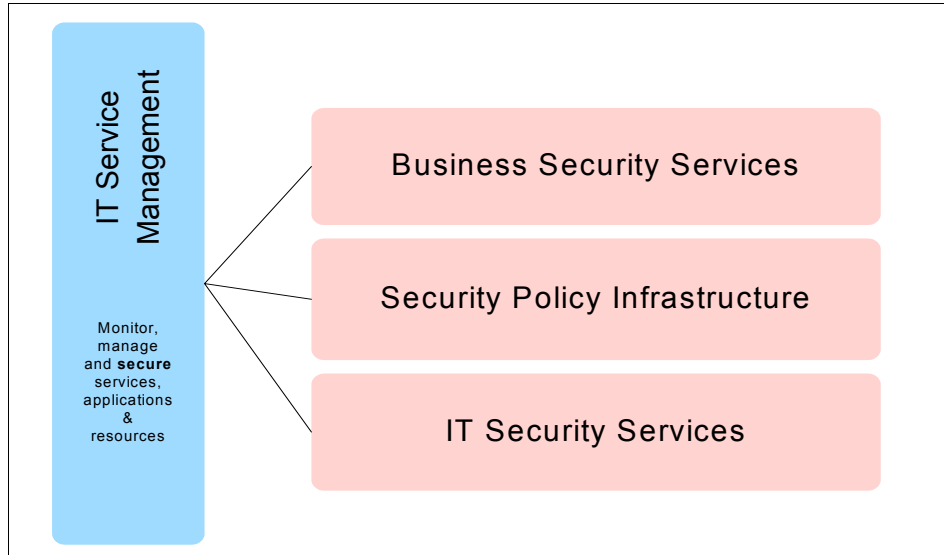


Figure 2-6 IBM SOA Reference Model - Security capability within IT Service Management.

At a very high level, we can define the following:

- ▶ *Business Security Services* leverage IT security services and Policy Infrastructure to provide business specific security capabilities.
- ▶ *Security Policy Infrastructure* deals with policy life cycle management specific to security; policies are enforced by security services, intermediaries, and policy distribution and transformation.
- ▶ *IT Security Services* are the building blocks to provide security functions as services.

Figure 2-7 shows the details of the IBM SOA Security Reference Model.¹

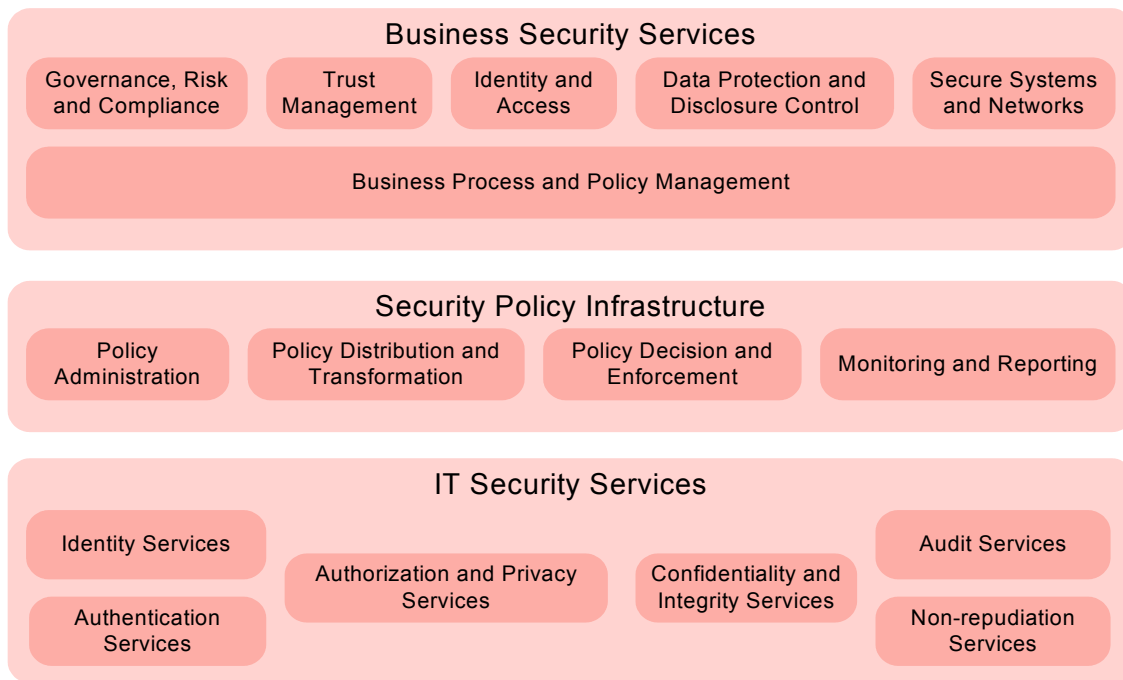


Figure 2-7 IBM SOA Security Reference Model

2.4.1 IT Security Services

IT Security Services can be used by different components in the SOA environment, including:

- ▶ Service consumers
- ▶ Service providers
- ▶ Gateways, proxy servers, and other intermediaries
- ▶ Application servers
- ▶ Data servers
- ▶ Operating systems

The use of common IT Security Services enables a consistent security implementation. It also minimizes development and deployment costs for these

¹ This is the first release of reference model derived by the Software Group Architecture Workgroup on SOA Security.

security services and for the SOA environment on which these security services are re-used.

The IT Security Services can be used like a blackbox; they should allow for integration with different authentication mechanisms, user registries, authorization engines, and so on.

Figure 2-8 shows the IT Security Services from the IBM SOA Security Reference Model. We will discuss each of these services in detail.

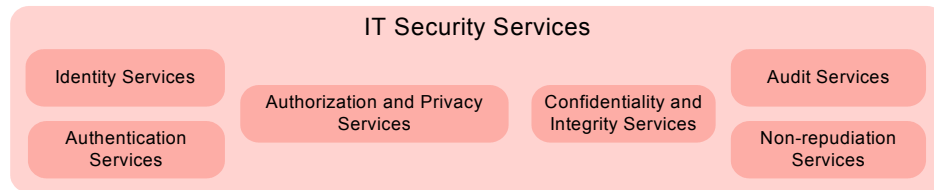


Figure 2-8 Security Services from the SOA Security Reference Model

Identity Services

The Identity Services are an abstraction layer and framework that provide for *managing, sharing, federating, and accessing* identity information from a variety of authoritative identity sources. These services also manage relationships between identities and provision identity information to multiple identity systems. The components within the Identity Service include:

- ▶ *Identity Foundation*: This component provides the uniform abstraction layer and administration facility needed to manage, store, and use the information about organizational entities (users, groups, roles, and so on), and to provide secure access to such information. It manages relationships between identities.

Identities are stored in user repositories. There may be a number of these in an organization, including a central user repository and separate user repositories associated with individual systems. Synchronization of identity information across these user repositories may also be required.

- ▶ *Identity Provisioning*: This component provisions and deprovisions identity information across multiple user repositories. Identity provisioning systems can implement provisioning policies, to ensure that identity information across a wide range of user registries is consistent with that policy. Alternatively, request based provisioning can be used to suit different business models. A combination of policy and request based provisioning may also be used.

When these systems cross trust domains, such as organizational boundaries, federated provisioning provides the capabilities to provision attributes and

other identity information across these boundaries. Standards based federated provisioning protocols are then the desired choice.

- ▶ *Identity Federation*: This component can provide identity relationships and mapping to help transform identities across trust domains. This is necessary for service requests to traverse security domains and be able to flow identity context as part of an end to end transactional flow.

Authentication Services

The Authentication Services provide capabilities to authenticate users to the system. These services may support multiple authentication mechanisms, such as user name/password, hardware token based, or biometric based. They may also support protocols such as Kerberos.

The Authentication Services also provide support for *identity tokens* and *security tokens* carried in messages, for example, Web services messages. Examples of these tokens can be SAML assertions and user name tokens. For example, the Authentication Services may call on a Trust Service for validation of authentication credentials within security tokens, or issue new security tokens with authentication credentials.

Authentication may be required at both the service consumer and service provider. Users may be requested to present authentication credentials at the service consumer to verify their identity to the environment. In this case, a security token may then be sent as part of the transaction flow from the service consumer to the service provider. The service provider authenticates the user based on this security token.

Alternatively, if the user has been authenticated at the service consumer, then an authenticated identity may be presumed, where the service provider trusts that authentication has already taken place. The service provider then accepts the authenticated identity carried in the identity token, without requiring another authentication. The binding of a SAML-based identity token to a request is one means of asserting an already authenticated identity.

Federated single sign-on can leverage both Identity and Authentication Service and allows a user to authenticate once to the federation, and by passing around security tokens, can be provided access to other trusted domains.

Authorization and Privacy Services

Authorization follows authentication. That is, once a user or system has been authenticated, it is then possible to perform authorization. Authorization means making a decision about whether an authenticated identity is allowed to access a resource. An authorization decision is dependent on two key inputs:

- ▶ An *authorization policy* that describes the required security attributes of a user or system that will allow them to access a resource.
- ▶ An *authenticated user or system* and their list of security attributes.

To make an authorization decision, policies need to be in place. These policies are enforced by a *Policy Enforcement Point* (PEP) that relies on the decision made by a *Policy Decision Point* (PDP). An example of a PEP is the Enterprise Service Bus, which would allow access to services based on the authorization decisions received from the relevant Policy Decision Point.

Note: The definition of decision and enforcement points is described in the International Organization for Standardization's standard 10181-3 Access Control Framework at <http://www.iso.ch>.

Privacy authorization is used to indicate the runtime function of authorizing access to *Personally Identifiable Information* (PII) based on a privacy policy. Hence for privacy, the granularity of authorization (for example, to personally identifiable information in medical records) can vary, and management of these policies would likely involve users as well as administrators.

Programming Model:

From a security perspective, the programming model includes decisions to be made about which components are responsible for enforcing security policies (for example, infrastructure or application) and what of information needs to be made available to requesters. In addition to the operational aspects, some of the design-time policy (for example, captured in J2EE™ deployment descriptors) can help manage the application. One of the key implementation decisions is whether the business needs will best be met by enabling the infrastructure to implement the security model or by codifying security enforcement into each application.

There are two common approaches to how security decisions are made:

- ▶ **Programmatic:** Decisions are made by application developers invoking APIs to make policy decisions. Application developers typically require deeper knowledge of the security system (or have to implement their own) when using this approach.
- ▶ **Declarative:** Decisions are made by the application server or a subordinate without the application developer needing to write any code. Security policy is written in a meta-language and accompanies the application, such as in a J2EE deployment descriptor.

In modern application development, the trend is towards increasing the use of *declarative security* to provide separation of duties between application developers and security administration, allowing each to focus on their areas of expertise. There are also three broad approaches on where security decisions could be made:

- ▶ *The application:* Application developers implement security logic alongside business logic.
- ▶ *The application server:* The application server uses its native policy configuration and evaluation constructs to evaluate policy decisions.
- ▶ *An external security provider:* In this case, the application server itself (for declarative security) or the application (programmatically) invokes a third party to evaluate policy decisions.

We normally recommend that applications focus on business logic, and defer securing the service access and the messages to the infrastructure (the runtime container hosting the application or external security providers invoked from the runtime container). In this infrastructure-managed approach, security policies attached to design artifacts are transformed into platform-specific policies (for example, requirements expressed via a UML model are transformed into J2EE deployment descriptors).

Use of external security providers rather than application or application server capabilities delivers the advantage of greater reuse of policies and provides the mechanisms for consistent security policy management and evaluation.

Confidentiality and Integrity Services

Confidentiality is the key security service for ensuring *non-disclosure* of sensitive information travelling on untrusted communication networks.

Confidentiality services protect information from unauthorized disclosure. This service provides protection of information in both storage and transmission. Requirements for Confidentiality Services, which exist at every level in the processing structure, depend on the granularity of protection necessary.

Integrity is the key security service for detecting unauthorized modification of data due to errors or malicious attack.

Integrity Services provides detection of the *unauthorized modification* of data. Organizations must allow for the use of data by authorized users and applications, as well as the transmission of data for remote processing. Data integrity facilities can indicate whether information has been altered.

Both Confidentiality and Integrity Services rely on cryptographic techniques. Encryption, message integrity codes, message authentication codes, digital signatures, and nonces all play a part in these services.

There are two additional sub-services integrated within the Confidentiality and Integrity Services protecting data at rest and data in transit.

Data Protection Services are concerned with data at rest. This includes security information and application information. For example, protection of cryptographic keys, passwords, and PII are all important. Encrypting the data in databases and signing audit information are additional examples.

Beyond cryptography, additional services for data protection include data and application isolation support provided in hardware, operating systems, and middleware.

The *Message Protection Services* are used to protect the message in transit from being:

- ▶ Disclosed (message confidentiality)
- ▶ Modified without detection (message integrity)
- ▶ Sent from a masquerading party (message origin authentication)
- ▶ Replayed (uniqueness)

This is usually achieved by encrypting or digitally signing a combination of the message body (or its parts), and header (or its parts).

Protection can take two forms:

- ▶ *Transport level protection*: Normally the whole data stream is protected at a protocol level below the application level. Secure Sockets Layer (SSL) is the most common example of a transport level scheme.

The disadvantages of this approach are that data is unprotected in intermediate nodes and this type of protection does not allow selective protection of message content.

- ▶ *Message level protection*: The actual message, or some parts of it, are protected at the application level. Therefore, even if the message passes through intermediate nodes, it is still protected.

Audit Services

Audit logging is the process of maintaining detailed, secure logs of critical activities in a business environment. Such critical activities could be related to security, content management, business transactions, and so on. Examples of security-related critical activities that could be audited are: login failures or successes, unauthorized or authorized access to protected resources, modification of security policy, noncompliance with a specified security policy, health of security servers, and so on.

An audit logging service provides mechanisms to submit, collect, persistently store, and report on audit data submitted as events. The events may be in a common format, such as *Common Base Event* (CBE²).

Which events are audited and stored is defined in an auditing policy. This policy should define which events are important, how long to keep the data, and if to keep the audit data in a tamper resistant form.

Audit data should be collected for all the security services and service providers.

Non-repudiation Services

Non-repudiation Services provide proof of *data origin* and *delivery*. They aim at preventing parties in a communication from falsely denying having taken part in that communication; for example, a non-repudiation service for digital certified mail should ensure that the sender cannot deny sending the message, and the receiver cannot deny receiving it.

² More information about CBEs can be found here:
<http://www.ibm.com/developerworks/library/specification/ws-cbe/>

The Non-repudiation Service should be able to:

- ▶ Protect a recipient against the false denial by an originator that the data has been sent
- ▶ Protect an originator against the false denial of a recipient that the data has been received

The Non-repudiation Services are implemented by cryptographic mechanisms that provide the following functions:

- ▶ Proof of origin of data
- ▶ Proof of delivery of data
- ▶ Proof of submission of data
- ▶ Proof of transport of data

The Non-repudiation Service is different from the Confidentiality and Integrity Services (per the ISO 7498-2 guideline). Non-repudiation is critical for *Electronic Data Interchange* (EDI) security and thus is a strategic element of the model. The digital signature mechanism is the principal implementation of a Non-repudiation Service.

Standards

Table 2-1 lists the security services and shows some standards and technology applicable to them. These will be mentioned throughout the IBM Redbook.

Table 2-1 Security Services standards

Security Service	Standards
Identity Services	SPML, SAML, WS-Federation, Liberty, and Identity Attribute Service (IdAS)
Authentication Services	WS-Trust, Kerberos, SAML, and PKI
Authorization and Privacy Services	XACML, JACC, WS-Authorization, and IDMix
Audit Services	CBE and WS-Base-Notification
Confidentiality and Integrity Services	WS-Security, WS-SecureConversation, PKI, XKMS, WS-SecurityPolicy, SSL/TLS, JSSE/JCE. XML DSig, and XML Encryption
Non-repudiation Services	PKI and ISO/IEC 13888:2004

For more information about security standards, see Appendix C, “Security standards and technology” on page 371.

2.4.2 Security Policy Infrastructure

Effective management of security policies requires a holistic approach that manages security policies throughout the life cycle of applications. Policies in the context of SOA are the means by which processes and services express the conditions and manage the behavior of the underlying infrastructure, in order to secure access to information, information availability and retention, audit, and so on.

Policy management includes authoring business policies that gets refined to service specific policies like security, performance indicators and metrics, trust policies, and so on. As shown in Figure 2-9 on page 39, these policies in turn get enforced by the infrastructure when they get configured as requirements that the infrastructure should meet.

Policies are defined and managed centrally. When there is a *Service Registry and Repository* (SRR) in the environment, then the *Security Policy Management* obtains service definitions and meta data from an SRR and defines policies based on that information. Once the effective policy is obtained, the policies are distributed to the *enforcement points*. Policies are distributed in a common format (Web services policy standards) from the Security Policy Management to the enforcement points. When there is a Service Registry and Repository in the environment, the Security Policy Management will publish policies to the SRR so that the enforcement points derive service definition and policies, and enforce them locally.

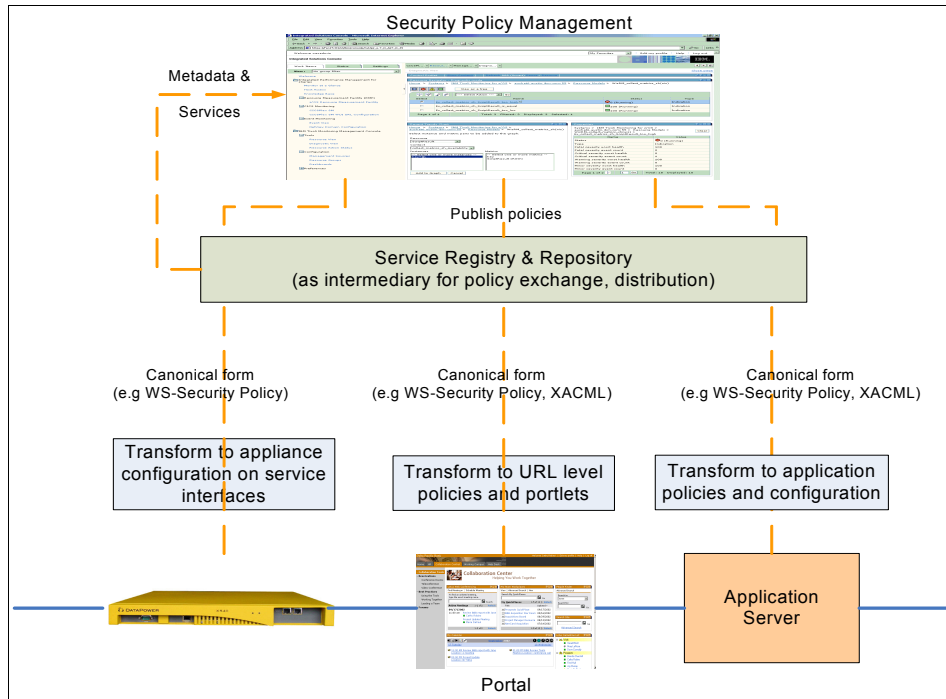


Figure 2-9 Security Policy Management: Definition and Enforcement

Figure 2-10 highlights the Security Policy Infrastructure within the IBM SOA Security Reference Model.

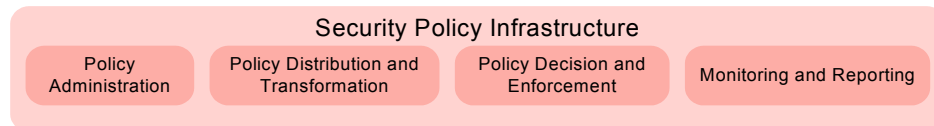


Figure 2-10 Components of Security Policy Infrastructure

Policy administration

Administration of the policies deals with the life cycle of a policy. This includes the activities of creating, managing, and importing/exporting the security policies using the security management tools available. There can be different kinds of policies. The following are two major categories:

Message protection policies

These policies focus on the capabilities and the constraints a service consumer needs to implement in order to invoke a service. The following items make up the message protection policies for the service consumer:

- ▶ *Service information*: The protocol to use to reach the service and the address of the service. For example, the service can be invoked using a Web service request over HTTP. If there are some intermediaries, the interaction policies between the service consumer and the intermediary also have to be known.
- ▶ *Message protection information*: Additional information about signing and encrypting of the message can be included in the policy; also, whether point to point or message level protection is required. This includes the encryption algorithms supported and the privacy rules that apply.
- ▶ *Identity information*: The type of security token required by the service provider (user name, SAML, Kerberos, or X.509 certificate).

Provider policies

The provider policies focus on the policies a specific provider applies based on its internal requirements. This can include authentication, authorization, privacy and audit policies. These policies are provided to the provider PDPs and enforced by the provider PEPs.

Taking an example, a service provider validates an incoming security token and performs authorization, ensuring, for example, that only account owners have access to their account information. This policy is checked based on the provided user information (user identifier and any other attributes from the user) and the information a user tries to access (the account number).

Policy distribution and transformation

The security policies created need to be distributed to the enforcement and decision points within the infrastructure with the appropriate information (see Figure 2-11 on page 41). The policies may be defined centrally and then are distributed to the enforcement points in a canonical format, for example, XACML, WS-Policy, or WS-Security Policy. The binding information to enforce the policies is also distributed appropriately. These policies are then transformed at the enforcement point to a local representation so that they can be enforced.

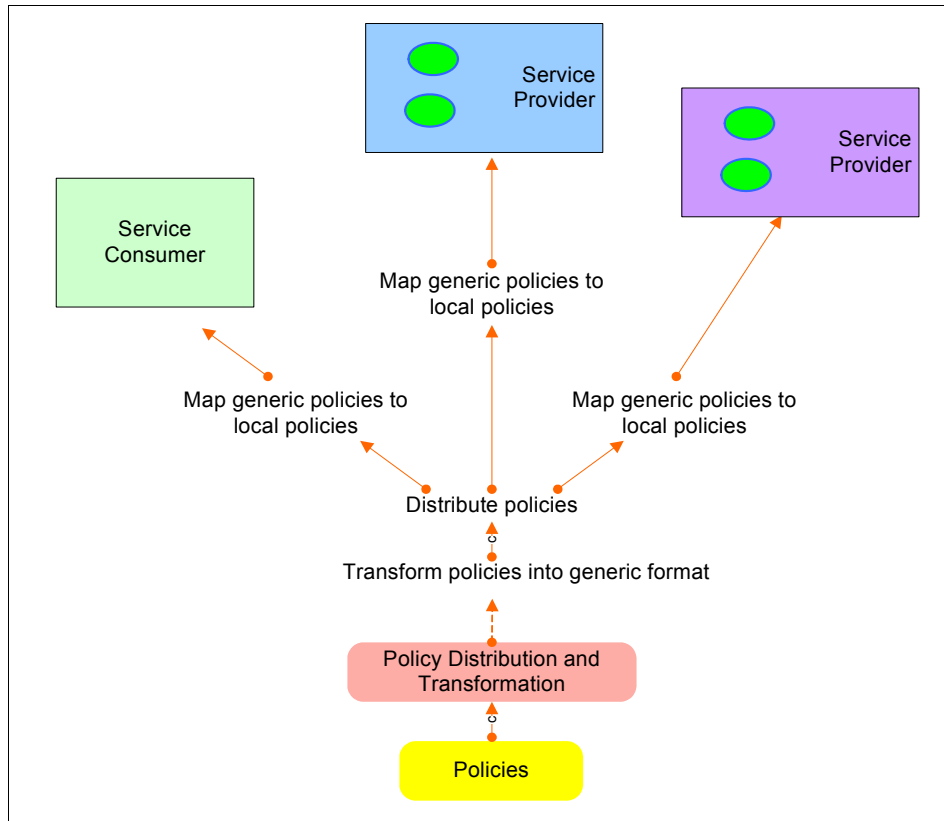


Figure 2-11 Policy Transformation and Distribution

Standards like WS-Policy and WS-SecurityPolicy provide descriptions on how service consumers and providers can specify their requirements and capabilities in a Web services world. Policy assertions can be defined for use within SOAP messages. Assertions can cover the authentication schemes (the required security tokens and the encryption algorithms), the transport protocol selection, the privacy policy, as well as information related to the quality of service.

The OASIS eXtensible Access Control Markup Language (XACML) provides a markup language to specify access control policies. It can be used as a generic format to store policies although it also provides a request/response model (based on XML format) for enforcement and decision points. It can be used to provide policies to a service provider as defined in “Provider policies” on page 40.

Note: More detailed information about the standards are available in Appendix C, “Security standards and technology” on page 371.

Note that there may also be a policy that controls the transformation and distribution of the policies.

Policy decision and enforcement

At runtime, there are two distinctive components of policy:

- ▶ *Policy Decision Point (PDP)*: A PDP makes a decision based on the policy and request. So, for example, it may consult an authorization policy to see if the user is permitted to access a service. The decision is provided to the policy enforcement point.
- ▶ *Policy Enforcement Point (PEP)*: Access to a resource is controlled by an appropriate resource manager or PEP. The PEP implements access control after making decision requests to PDPs using, for example, XACML (Appendix C, “Security standards and technology” on page 371).

In a typical deployment, you can find several enforcement points. Each of these enforcement points can have its own mechanisms to enforce security for the incoming requests.

The enforcement points rely on *decision points* to make decisions. These decision points contain the security policies defined in the infrastructure. The requirements and thus the policies are different, depending on the security domains and the application platforms.

- ▶ The *service consumer* can enforce a first level of security on the client side. Security controls based on the policies provided for this service can be made so that the request matches the requirements.
- ▶ The *service provider* can enforce security policies for incoming requests and provide a first level of defense for the infrastructure. These include enforcing message protection, provide token validation and exchange, as well as authorization and audit services as defined in the security policy.

One of the challenges of enforcement and decision points is to have multiple enforcement points within the infrastructure talking to several decision points. Providing integrated and centralized decision capabilities reduces the administrative tasks related to policy management.

Monitoring and reporting

It is necessary to keep track of applicable policies, any modification to them, as well as the assessment of compliance of lower-level policies against corporate policies. Traceability of policies from high level business policies to enforced configurations and runtime requirements is necessary to track what the goal is and what the runtime behavior is based on. This helps identify policy changes that occurred and can help manage accountability of policy changes. Closely linked to this practice is the enforcement of these policies in the infrastructure and the monitoring of the behavior of system elements throughout the life cycle of the business. Security policies need to be developed and deployed throughout the stages of the life cycle of an application.

Once an application is made available to be accessed, application related policies get administered to reflect any changes that may happen during the lifetime of the application. Changes to security policies include authorization policy changes (for example, adding new roles that may access the resources or assigning roles to new user groups or users), user management changes (for example, users assigned to additional user groups), or other changes, including audit requirements, and constraints like integrity or confidentiality.

When administering security policies, it is necessary to adhere to changing corporate business security policies and industry and government regulations, and compliance requirements. In addition to these sources of change, another key input factor for change in policies is the discovery of vulnerabilities and new risks that may be identified through solution monitoring activities. These changes to security policies must be tightly controlled and access to them should be traced and audit trails supplied so that the processes may be adequately monitored.

2.4.3 Business Security Services

In addition to securing business services, it is necessary to provide a secure deployment environment where business solutions can be deployed and hosted. Business Security Services are depicted in Figure 2-12.

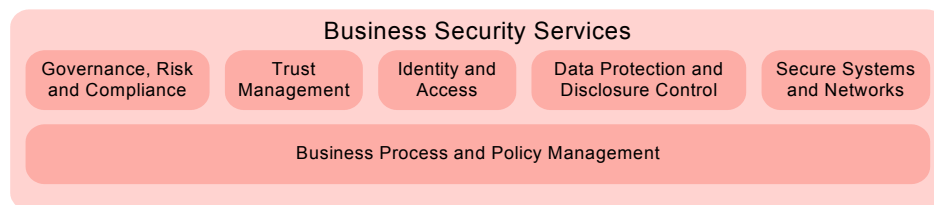


Figure 2-12 Business Security Services

Governance, risk, and compliance

Governance of SOA Security is a subset of the overall SOA Governance function. Governance is very important for the security services, as managing the security policy and implementation is vital to the integrity of the environment.

A framework towards effective governance structure and decision making authority is needed in order to run the business. Tools and technologies can help facilitate governance initiatives and compliance evaluations. An effective security governance framework involves establishing chains of responsibility, authority and communication to empower people to effectively control the system.

Because SOA extends interactions beyond the enterprise boundary, the governance of SOA Security must interact with similar groups in other organizations to achieve a common set of standards for communication across the enterprise boundary.

Risk management deals with the process of evaluating and assessing risk in the SOA environment, and developing strategies to manage those risks. Risk management is a cost-benefit exercise; it is not feasible to eliminate all risks in an SOA environment. The risk management process determines how to manage risk based on factors, such as probability and impact. While software tools can help implement a risk management process, people and processes are the main components.

Compliance management measures the performance of the IT system relative to the measures established by the business policies. This might include verifying the working system against a set of internally created policies, and also against external Federal or State regulatory acts.

Audit records form the basis of the raw data required for compliance assessments. The compliance function described in this section may be a manual process, or an automated tool could be used to reconcile the business compliance requirements with the raw data extracted from the audit service.

Managing the audit data involves assessing the implementation of the security elements of the SOA solution against the solution design. You might also attempt to identify inconsistencies between the configuration of multiple instances of a solution component that should share an identical security configuration. A third aspect is the verification of the configuration of the security services themselves. Periodic auditing of the components and overall SOA solution are recommended.

Trust Management

Trust Management addresses trusted relationships between entities like organizations, enterprises, identities, security domains, and systems. These relationships can be system-to-system, business-to-business, and so on.

Trust Management deals with two aspects, namely business and technology. The *business aspect* deals with two entities agreeing upon a set of rules to conduct business. These rules include relationship management, liability management, and other legal aspects.

The *technology aspect* deals with managing the infrastructure that supports the capability for establishing trust by cryptographic methods. These include key management (strength, key validation, and so on) protocols, attributes, and other technical considerations for establishing trust.

There are multiple ways of establishing trust relationships. In Figure 2-13, the trust may be explicit and simple, where consumer and provider are within a single trust domain, and thus have the same trust source.

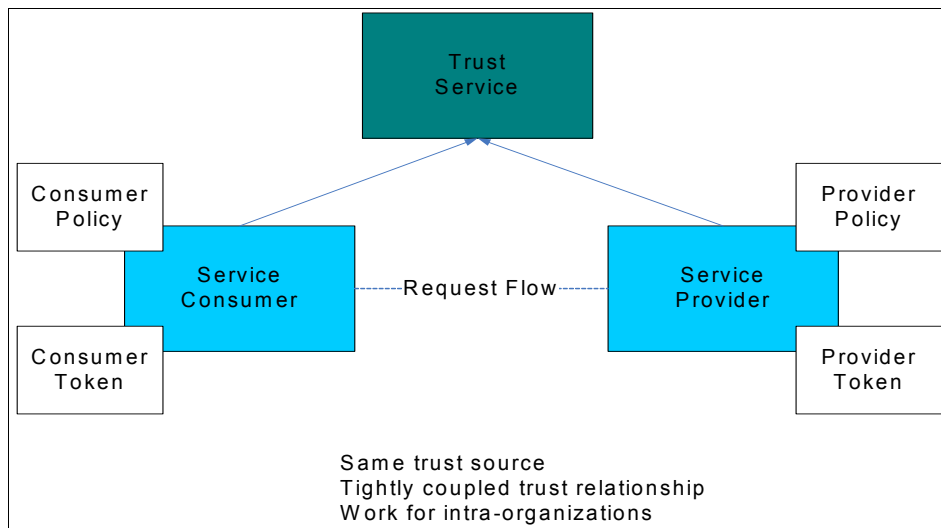


Figure 2-13 Tightly coupled trust relationship

In Figure 2-14, we illustrate another approach where a consumer and target service may have separate trust zones and trusted relationship, or a trusted third-party trust service.

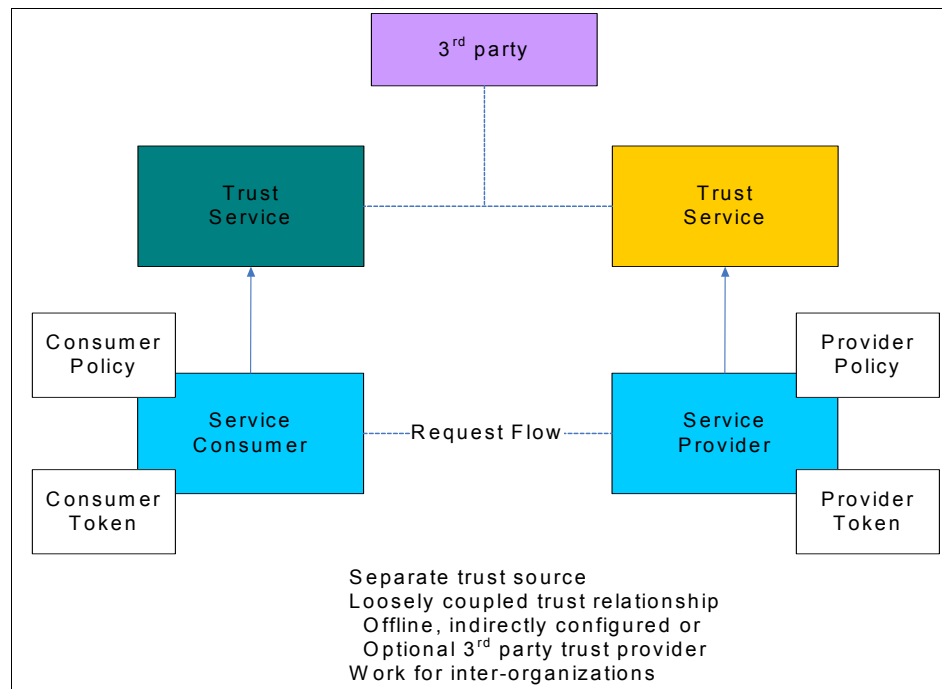


Figure 2-14 Loosely coupled trust relationship

Identity and access

This deals with the technologies that are needed to manage identities both within an enterprise as well as across enterprises. It also includes management of access policies to resources based on identity information and resource information.

Identity life cycle management is the main task. Identities need to be created, modified over time, and eventually deleted. Some important aspects are:

- ▶ *HR identity feed*: Often the authoritative source of identity information for internal users is the HR (human resources) system. An identity feed from the HR system can indicate, to the identity management system, that changes to the user population have occurred, and provisioning workflows need to be initiated.
- ▶ *Approvals*: Before accounts on end systems are created or modified, approvals from the appropriate management may be required. This can be automated.

- ▶ *Re-validation*: Access to systems may need to be approved at regular intervals. The system should collect the appropriate re-validation approvals.
- ▶ *User self-care*: Users of the system should be able to perform certain tasks without input from an administrator. For example, they may want to self-enrol to the system, reset or change their password(s), and so on.
- ▶ *Delegated administration*: For approving requests for accounts, and other administrative functions, delegating the action to another user or users is an important function.

Data protection and disclosure control

Data protection management deals with protecting business information and provides the capabilities for content and data protection in transit and at rest. It includes policies for which data is to be protected and to what extent it can be specified and implemented. Externalizing data handling rules from applications and IT systems can help to simplify the management of data protection.

In the context of information and business information privacy, the disclosure control capability helps reduce privacy compliance costs by automating manual procedures. The system builds trust by:

- ▶ Publishing a privacy policy for users to view
- ▶ Managing user consent to privacy policies
- ▶ Capturing user preferences (such as opt-in to release of PII for certain purposes).
- ▶ Getting detailed reports on access to sensitive information

Secure systems and networks

This is a category of technologies and embedded systems that help protect infrastructure servers, systems, and networking resources from security threats. The desire is to protect the systems from external and internal threats, such as hackers and viruses.

Firewalls are used whenever there is a need to control the traffic between two networks. For example, a firewall is used at the connection of an organization and the Internet, and may provide simple protocol and port filtering, or more complex protocol inspection. Newer types of firewalls inspect XML and SOAP traffic and provide protection against higher-level protocol specific attacks.

Operating system security involves *hardening* of commercial operating systems so that they provide greater security controls. For example, one issue with UNIX operating systems is that the administrative user (root) has full control, including deleting all security audit logs. In this case, operating system security software can control and securely log the access of root to applications and data, providing separation of duties.

Intrusion detection (host and network) is concerned with detecting anomalies in the use of the operating systems or the network. This might be used, for example, to detect external or internal intrusions to these systems.

Virus detection is used to detect and delete any viruses. This might be implemented at the border of the organization and the Internet, and also on individual host operating systems.

Patch management involves applying service patches to operating systems, application middleware, and databases and applications within the environment. These patches might contain security fixes that remove vulnerabilities in software.

Business Process and Policy Management

Business Process and Policy Management applies to all the Business Security Services and deals with coordination and integration of business processes to optimize and adapt their processes for maximum efficiency.

Some examples of these processes and policies are:

- ▶ Governance, risk, and compliance

Business processes and policies are needed for defining organizational roles and responsibilities for process and authority. Risk management processes and policies are needed to evaluate strategies for managing risk versus cost. Compliance may include assessment processes and reporting policies, for example, what type of assessment is used, how often it is executed, and who should be informed.

- ▶ Trust management

Business process and policies are required for establishing trusted relationships. These processes may include who to include in a circle of trust, what legal process to follow, and what process is used for evaluating liability. This may also include the policies for what type of access to resources.

- ▶ Identity and access

Processes for identity can include on-boarding and off-boarding identities, and self-care / self-registration for optimal user interaction. It can also include processes and policies for approval of access to IT resources and business

resources. In addition, policies for password management and identity management are also applicable.

- ▶ Data protection and disclosure control

Business policies are needed to define content and data for use in transit and at rest. Processes are needed in the event of misuse and handling inappropriate use of data. Business policies are also needed to define sensitivity of data and apply the appropriate message protection and privacy policies.

- ▶ Secure systems and networks

Policies are required for intrusion detection and event management for ensuring secure systems and networks. Processes must be in place for handling alerts, for engaging the Computer Emergency Response Team, and for normal housekeeping of scheduled maintenance, patch management, and servicing.

2.5 IBM SOA Security Logical Architecture

This section introduces some typical SOA scenarios as use cases where various aspects of security are needed and enforced. We use these scenarios to derive a logical deployment architecture and then apply the IBM SOS Security Reference Model to derive the IBM SOA Security Logical Architecture.

2.5.1 Foundation scenarios

IBM SOA Foundation scenarios are a group of reusable assets that can help speed up the process of adopting SOA. This section briefly describes three of the IBM SOA Foundation scenarios and some of the security issues involved. More details are included in the following chapters.

Service Creation scenario

This scenario depicts a typical situation where existing business logic (likely in the form of an application) is to be enabled as a service, or a new service is to be deployed. For example, access to business logic from an enterprise information system like CICS would fall in this category. In such cases, a service veneer may be built so as to allow access to an existing business application. For example, one may build a Web service implementation using a J2EE application and provide this veneer implementation. In such cases, an application server that provides a SOA runtime environment could host this new service that publishes the capabilities using a service interface. Security is an important consideration in this case. It is not only important to protect the message but to identify who is invoking the service and be able to record that for accountability. It is also

important to provide a seamless user experience such that the intermediate veneer is transparent to the user.

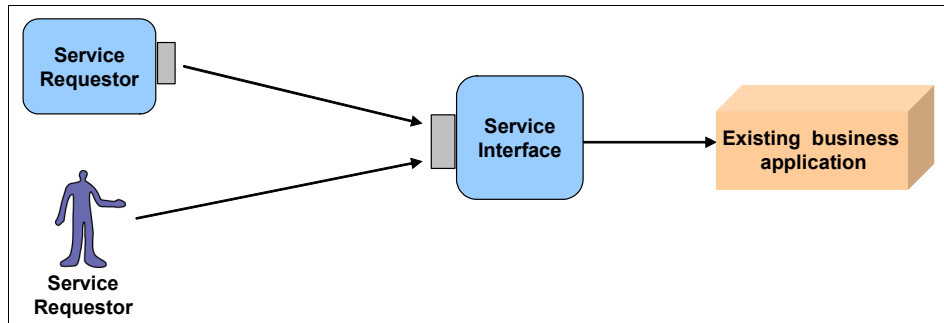


Figure 2-15 IBM SOA Foundation Scenarios: Service Creation

Securing access to such a service could be handled through declarative security means so that the application does not handle security, but lets the service hosting runtime (application container) provide the necessary security enforcements. Security services like authentication, token mediation, identity mapping, confidentiality and integrity, and auditing may need to be leveraged to provide secure access to the service. In addition, we may also need SSO for a seamless user experience.

For more information, see Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61, which focuses on applying the IBM SOA Security Reference Model and Architecture to the Service Creation scenario.

Service Connectivity scenario

There are situations where an enterprise has a set of core services or systems that are to be made available as services to a variety of internal and external systems and users. The flexibility to make changes to their service and service implementations with minimal to no impact to service consumers is desired. In such cases, an *Enterprise Service Bus* (ESB) can provide the necessary decoupling of service consumers and service providers. As shown in Figure 2-16 on page 51, consumer's systems may also use different protocol bindings compared to what the services provide.

In such cases, it is important to protect business information and establish business trust relationships for identity, data, and so on. Since the requests may come from the external entities, security domains are likely crossed, hence propagation of identity across domains is also a key consideration. In addition, the enterprise must be able to subscribe to governance, risk, and compliance for a variety of legal and regulatory aspects.

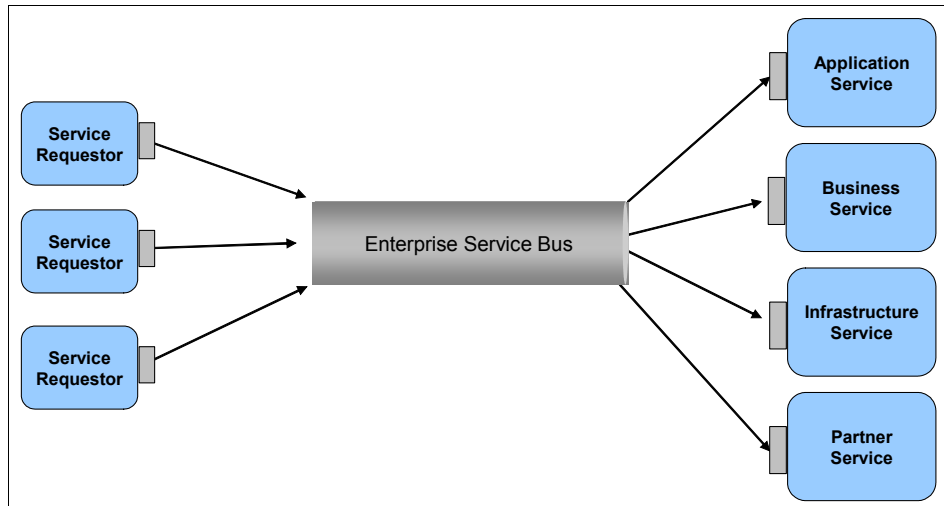


Figure 2-16 IBM SOA Foundation Scenarios: Service Connectivity

The ESB gateway would leverage the security services responsible for message level security, confidentiality and integrity, identity and authentication, authorization and privacy, federation of identities between external consumer and provider environments, and manage the trust relationship between the external consumer and service provider. The ESB may also call on the same services for requests from internal consumers as well. There may also be the same set of requirements for the establishment of trust relationships between internal consumer and providers that might be in different business units of the organization.

Chapter 4, “IBM SOA Foundation Service Connectivity scenario” on page 89 focuses on applying the IBM SOA Security Reference Model to the Service Connectivity scenario.

Service Aggregation for User Access scenario

This scenario discusses the aggregation of disparate services on an integrated consumer portal or employee workspace, in other words, *on the glass* integration for enhanced user experience and increased productivity through role based employee portals. These services can be either Web-based or Web services based applications that need to be accessed from an integrated workspace like a portal. Users may access this through a variety of client tools, including browsers, PDAs, kiosks, and so on. Furthermore, these services can be available from within the enterprise or external to the enterprise at a service provider company.

Figure 2-17 shows the scenario.

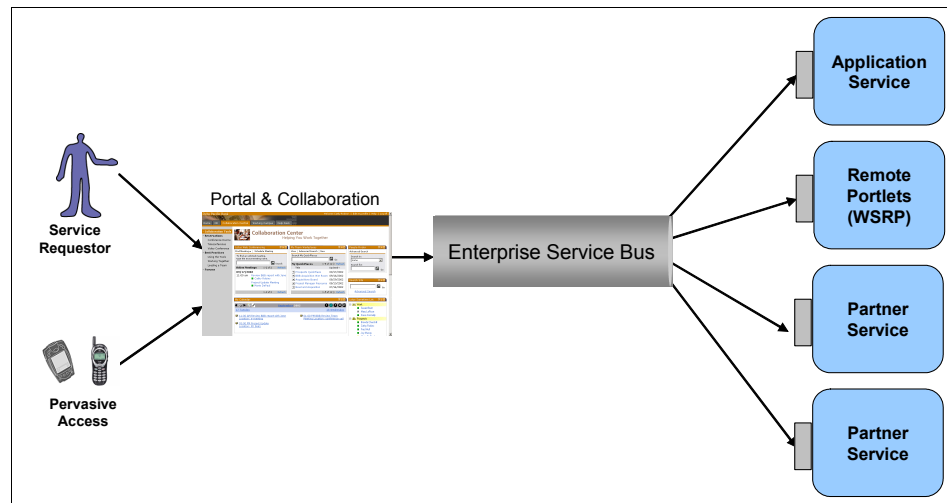


Figure 2-17 IBM SOA Foundation Scenarios: Service Aggregation

User experience via single sign-on to all the applications, self-care, and profile management are very important in this scenario. A user only needs to be identified/authenticated once, and then should be able to access all of the applications. The portal provides access to internal service provider applications. A service already exposed can be reused in this scenario. Securing the connection from portal to the internal service providers, providing the security tokens, and identity mapping may all be required.

In the case where the portal accesses external service providers, the use of identity federation techniques for security token and identity mapping are also relevant. The company portal can make requests to the external portals as well. For example, the Web Services for Remote Portlets (WSRP) standard may be used. The same security requirements exist as for the external service provider case. There is also the case where the user needs federated single sign-on to enable them to access the external portal from their browser without needing to sign in there again. Protocols such as SAML or WS-Federation may be used in this case, and again require the identity federation services. Since a number of application domains are being traversed, it is imperative to provide accountability for user's identity propagated to all applications and service providers.

Note: The complete specification for the Web Services for Remote Portlets is available at <http://www.oasis-open.org/>.

A number of security services can be leveraged to provide the capabilities discussed above: message level security, confidentiality and integrity, identity and authentication, authorization and privacy, federation of identities between external consumer and provider environments, and managing the trust relationship between the external consumer and service provider. There may also be the same set of requirements for the establishment of trust relationships between internal consumer and providers that might be in different business units of the organization.

Chapter 5, “IBM SOA Foundation Service Aggregation scenario” on page 117 focuses on applying the IBM SOA Security Reference Model to the Service Aggregation scenario.

2.5.2 Typical deployment architecture

Based on the scenarios in the previous section, Figure 2-18 on page 54 shows a typical deployment architecture. Explained in a simplified manner, there is usually a proxy (HTTP or Web services gateway) in the DMZ followed by either an application server or a presentation server. The application/presentation server leverages existing applications/services either directly or through an ESB. Clients can be users or service consumers, both internal to an enterprise or external. Similarly, existing applications/services can be either internal or external to an enterprise.

In such a deployment, security is enforced at various points within the architecture. The proxy may enforce confidentiality and integrity, identity validation, authentication, and auditing. The identity derived at the proxy may need to be propagated to the application server, where the propagated identity needs to be accepted and additional security checks like authorization and auditing may be enforced. Further security enforcements may be performed by the other components of the architecture as well.

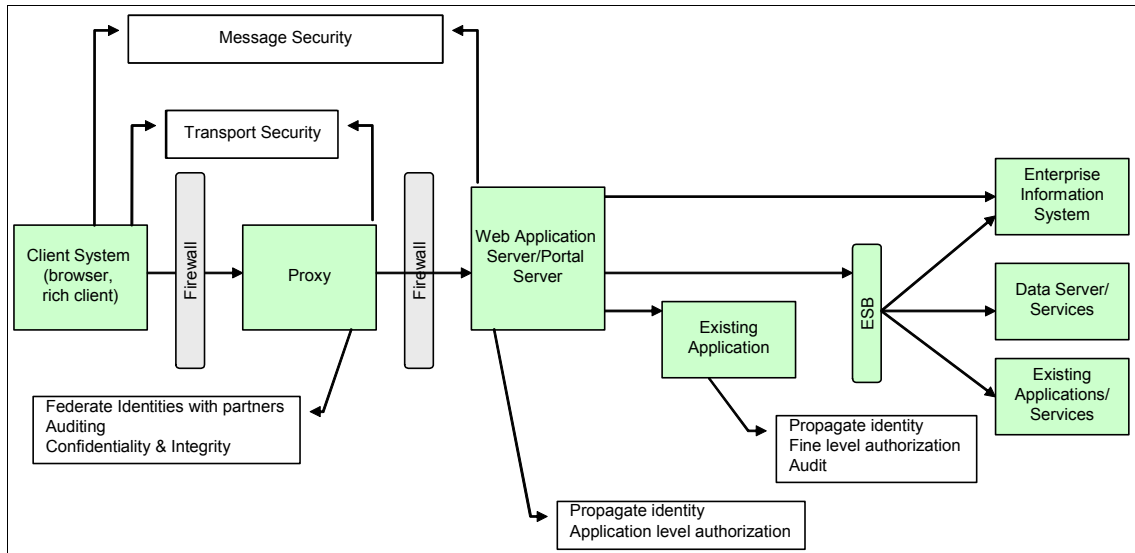


Figure 2-18 Typical deployment architecture for an SOA application

Two main observations can be derived from the above discussion:

- ▶ *Security infrastructure integration challenge*: Since each component is enforcing some aspect of security, there may be multiple identity and authentication systems, multiple authorization engines, and multiple audit logs. In a typical environment, these security systems are not well integrated and hence can be considered a challenge in a SOA environment.
- ▶ *Security management challenge*: From a management perspective, there are multiple islands of administration specific to products and usually prone to errors, inconsistency, and lack of coordination. Management is resource centric and policy management is isolated to a business unit or application or product. This makes it a challenge in a SOA environment where consistent policies have to be enforced across multiple components of the architecture.

The IBM SOA Security Reference Model addresses the above challenges by applying a consistent model for both policy management as well as policy enforcement by leveraging IT Security Services.

In Figure 2-19 on page 55, we address the problem of *identity propagation* by applying the IBM SOA Security Reference Model to the logical deployment architecture. In this example, we show that we can leverage the *Identity and Access* business security service to define the appropriate policies. These policies are defined and managed by the *Security Policy Infrastructure*, which will distribute them in a consistent manner to all the relevant components within a logical deployment architecture. These policies are enforced by security

enforcement points within these components by *IT Security Services*. These IT Security Services can be either available locally (like within a browser) or can be leveraged by centralized services (like the proxy taking advantage of external enterprise Identity and Authentication services).

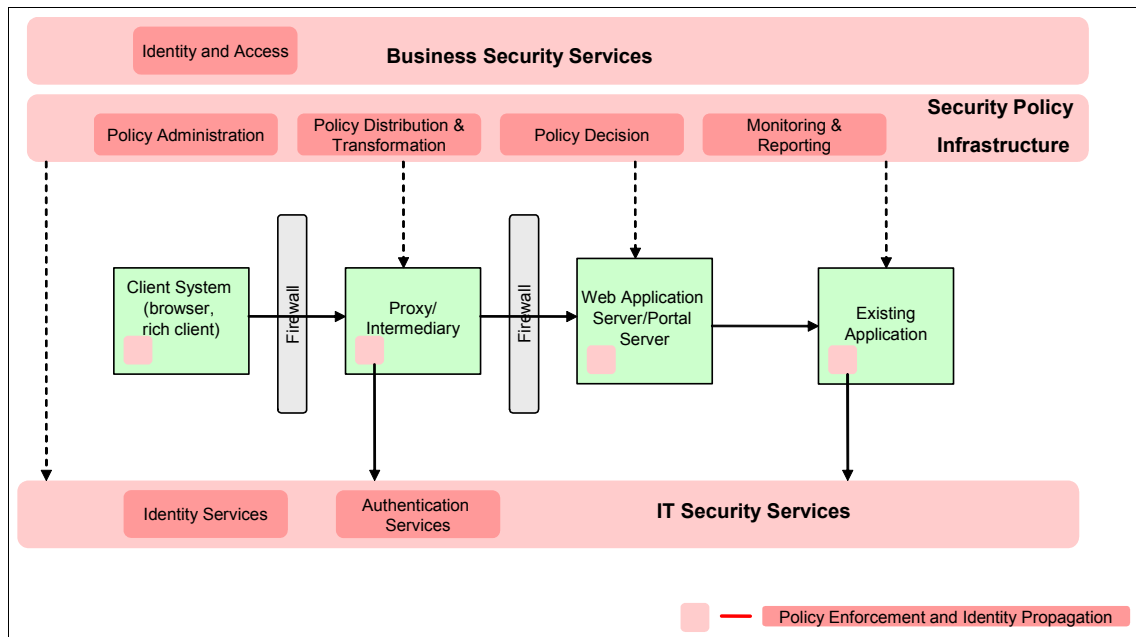


Figure 2-19 Identity propagation using IBM SOA Security Reference Model

Although the above example only demonstrates one particular security function, the same can be applied for other functions of security as well. The next section shows how we can make this generic and derive an IBM SOA Security Logical Architecture based on the IBM SOA Security Reference Model.

2.5.3 IBM SOA Security Logical Architecture summary

We can extend the above example to other SOA security requirements and apply the IBM SOA Security Model to a typical SOA deployment logical architecture to derive the IBM SOA Security Logical Architecture, as shown in Figure 2-20 on page 56.

In this architecture:

- ▶ *IT Security Services* are the building blocks to provide security functions as services.

- ▶ *Security Policy Infrastructure* not only provides security policy life cycle management but also policy distribution and transformation. Policies can be associated with service definitions and metadata and published back to service registries.
- ▶ *Business Security Services* leverage IT Security Services and Security Policy Infrastructure to build business specific security services.

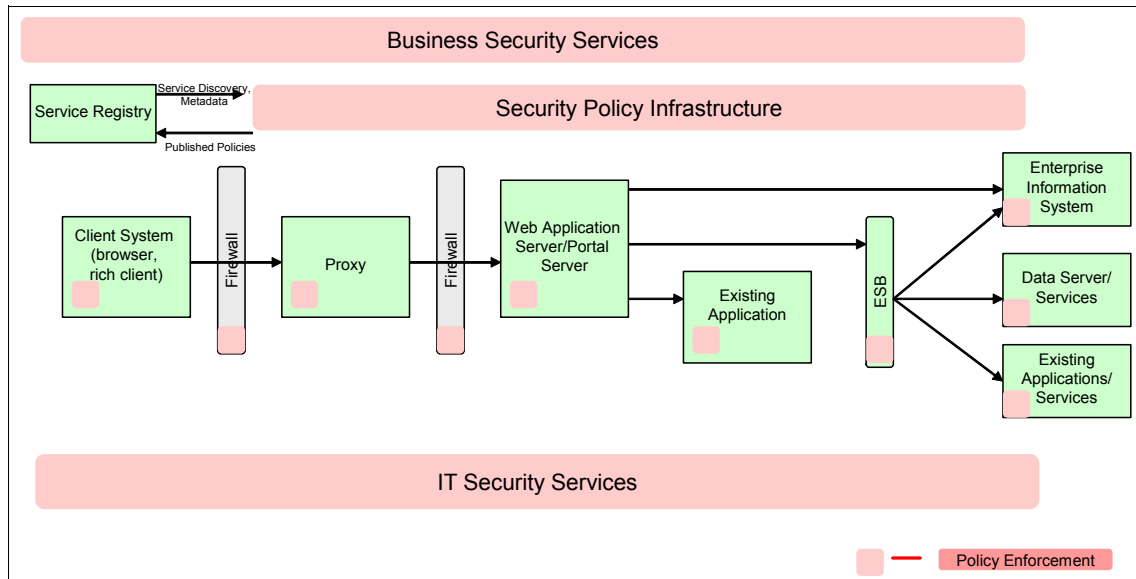


Figure 2-20 IBM SOA Security logical architecture

As we have seen before, there are multiple security enforcement points within a SOA environment. These enforcement points derive consistent, coordinated, business driven policies from the Security Policy Management. These policies are based on metadata that can be derived for Service Registries when available. Since the applications are shared/reused, the applicable policies to address changing needs, heterogeneous application platforms, and protocols (across organizations and vendors) are easily accommodated.

Let us repeat once more: Policies are distributed not only to different enforcement points but also to IT Security Services. These policies are enforced by security enforcement points within these components by IT Security Services. These IT Security Services can be either available locally (like within a browser) or can be leveraged by centralized services (like the proxy taking advantage of external enterprise Identity and Authentication services).

2.6 Conclusion

To enable an enterprise so that its processes and applications are flexible, one must start by expecting changes, both to process and application logic, as well as to the policies associated with them. Security must be factored into the SOA since security is a business requirement and not just a technology attribute. At the core of all SOA security lies a policy-based infrastructure and management of the policies. In the ideal case, the SOA application is centered on business logic, delegating the enforcement of security policies and the handling of trust relationships to the infrastructure. Security services are essential building blocks that can provide virtualization of security capabilities, as services themselves, so that different infrastructure components can consistently access security functionality. Thus, security services form the fabric for any SOA environment.



Part 2

IBM SOA Foundation scenarios

IBM SOA Foundation scenarios are a group of reusable assets that can help speed the process of developing SOA based applications. This part describes three of the IBM SOA Foundation scenarios and the security issues involved.



IBM SOA Foundation Service Creation scenario

This chapter describes the application of the SOA Security Reference Model, as defined in Chapter 2, “Architecture and technology foundation” on page 17, to the SOA Foundation Service Creation scenario. This scenario is the first of the IBM SOA Foundation scenarios. The chapter highlights how each component of the SOA Security Reference Model can be applied to the scenario.

3.1 Scenario overview

This section contains a brief description of the SOA Foundation Service Creation scenario.

The Service Creation scenario describes how to expose existing application functionality and new business logic as services. These services may then be consumed by other services or client applications within an enterprise and between enterprises.

For example, a customer has a core set of IT or business functions that offer value to a variety of internal and external clients. The customer wants to enable a wider use of these functions without incurring the complexity of point-to-point integration with each of its clients.

By exposing the application functionality as services, client applications internal and external to the enterprise can consume these services. This SOA approach simplifies the integration challenges and leverages the business value of existing systems.

3.1.1 Direct exposure architectural pattern

There are two main architectural patterns to illustrate the Service Creation scenario. Figure 3-1 on page 63 shows one pattern, where the existing Enterprise Information System (EIS) applications are directly exposed as services. In this architectural pattern, the service interface is defined largely by the existing application.

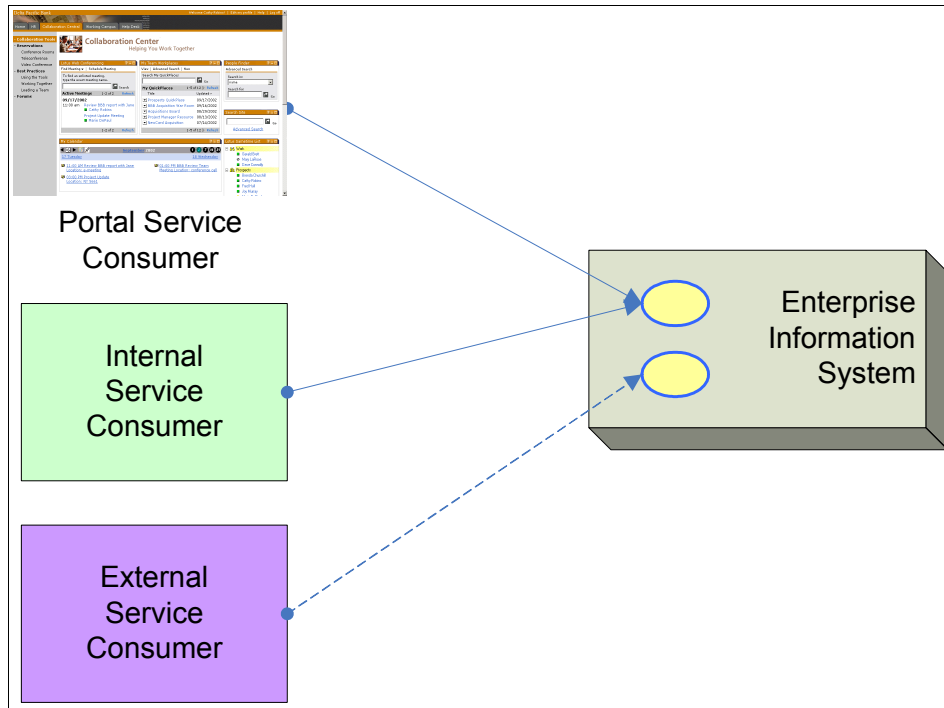


Figure 3-1 Directly exposing existing applications as services

As shown in Figure 3-1, many service consumers can invoke the services exposed from the existing applications. The benefits of this approach include a shorter deployment cycle through the re-use of the existing assets.

Examining the figure, we have three types of service consumers:

- ▶ User accessing via a portal: The user interacts with the portal via a browser. The portal generates services requests to the EIS on behalf of the user.
- ▶ Internal service consumer: An application generates service requests to the EIS. This might be from a user with a thick client application, or from another application within the environment that is not directly user driven.
- ▶ External service consumer: The difference from the internal case is that these service requests are from outside of the domain of the EIS. For example, the requests are from another organization, or another business unit of the same organization. The dashed line indicates the requests are coming from a foreign domain.

Figure 3-2 gives an example of the direct service to illustrate a real product implementation. In this case, the EIS is CICS Transaction Services (TS) 3.1, which has the capabilities of exposing its applications directly as Web services. The portal is WebSphere Portal, with its portlets generating Web service requests. The internal and external service consumer applications are WebSphere J2EE and Microsoft® .NET applications respectively.

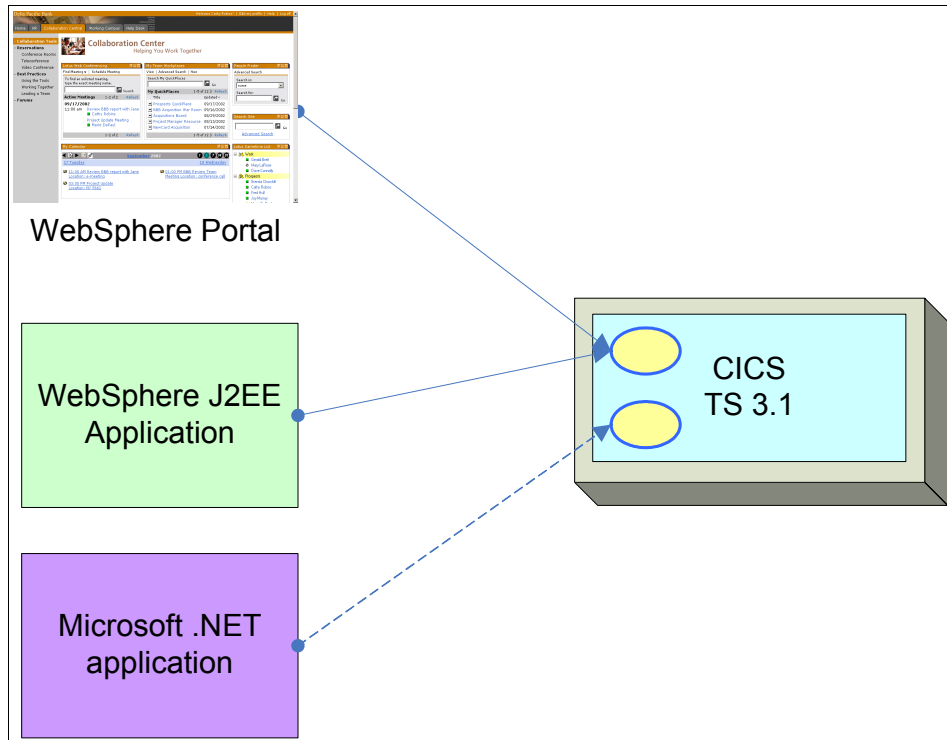


Figure 3-2 Directly exposing existing applications as services - product example

3.1.2 Indirect exposure architectural pattern

The indirect exposure pattern is shown in Figure 3-3. This illustrates how to indirectly expose the existing EIS applications using service components. In this architectural pattern, a middle tier is used to create services from the EIS application. This may be used, for example, when using several services from different sources. A middle tier may also be required when additional business logic is needed to bridge the interface between consumers and the set of directly exposed services.

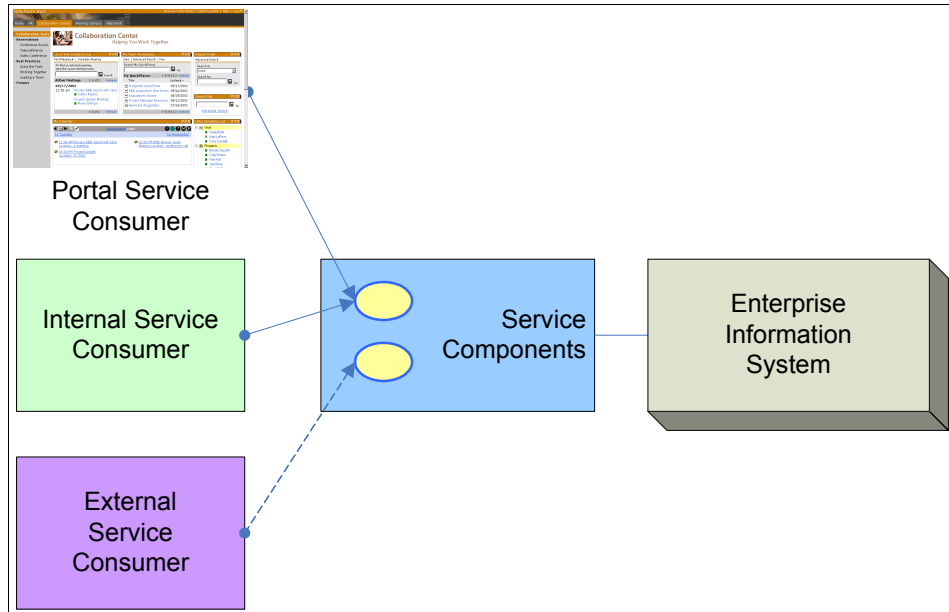


Figure 3-3 Indirectly exposing existing applications via service components

Figure 3-4 gives an example of the indirect exposure pattern to illustrate a real product implementation. In this case, the EIS is CICS Transaction Services (TS) 2.2. Services are exposed using WebSphere Application Server and the CICS Transaction Gateway (CTG). The portal is WebSphere Portal, with its portlets generating Web service requests. The internal and external service consumer applications are WebSphere J2EE and Microsoft .NET applications respectively.

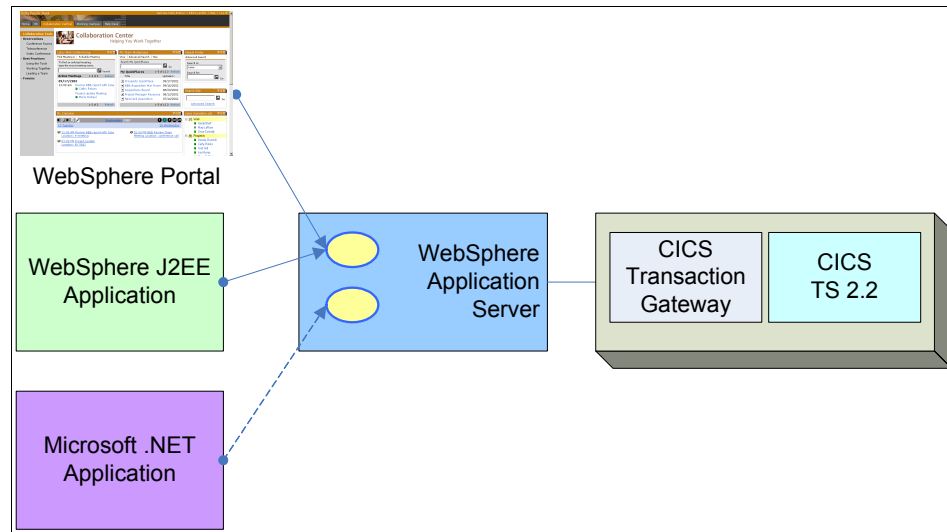


Figure 3-4 Indirectly exposing existing applications via service components - product example

3.2 Applying the IBM SOA Security Reference Model

Figure 3-5 on page 67 shows the SOA Security Reference Model. There are three main components to the reference model: IT Security Services, Security Policy Infrastructure, and Business Security Services. A more detailed description of each of the components is included in Chapter 2, “Architecture and technology foundation” on page 17.

A key point to understand when reading this chapter is that it describes how each of the SOA Security Reference Model components *could* be applied to the Service Creation scenario. The decision about which components *should* be applied for a particular implementation of the Service Creation scenario depends on the relevant business requirements. For example, in some cases the services have high security requirements, such as banking or insurance. In other cases, the services have lower security requirements, such as public information sources. An IT architect must therefore decide which of the SOA Security

Reference Model components should be applied for their particular implementation.

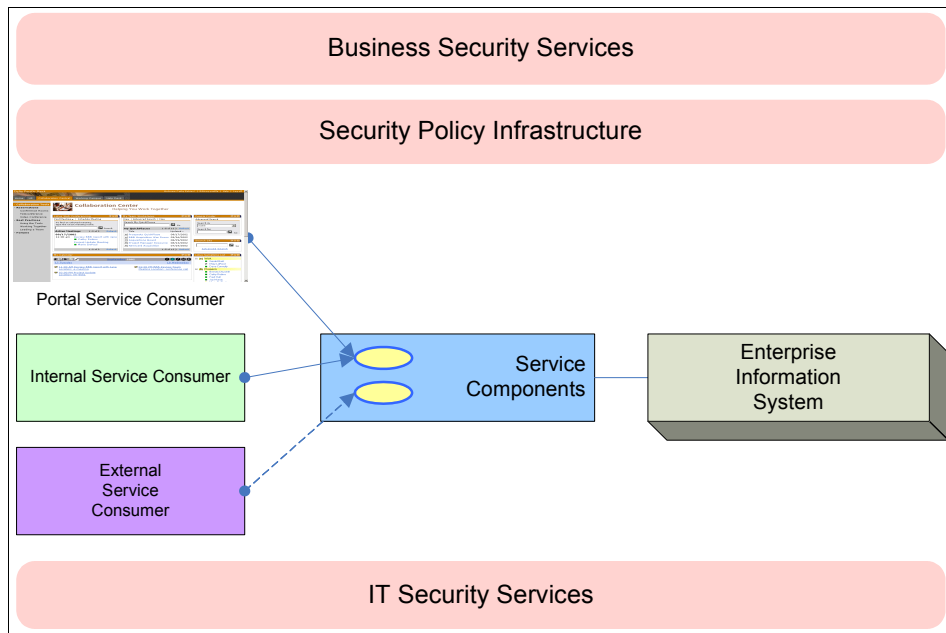


Figure 3-5 Applying the SOA Security Reference Model

3.2.1 IT Security Services

IT Security Services can be used by different components in the SOA environment, such as gateways, proxy servers, application servers, data servers and operating systems. The use of common IT Security Services enables a consistent security implementation. It also minimizes development and deployment costs for implementing these services.

Figure 3-6 shows the IT Security Services from the SOA Security Reference Model. We will discuss each of these security services in turn, and show how they can be used within the SOA Foundation Service Creation scenario.

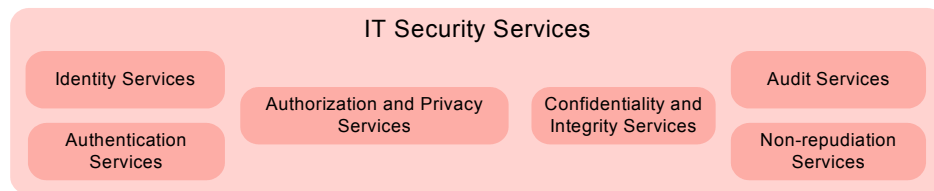


Figure 3-6 IT Security Services from the SOA Security Reference Model

Identity Services

In an SOA environment the most fundamental security issue to deal with is often related to the Identity Services. Figure 3-7 shows the direct architectural pattern of the Service Creation scenario showing a typical identity environment.

Identity foundation

There are multiple user repositories required in Figure 3-7. This is for a small environment and indicates the problems faced. In a typical client environment, there may be many more user repositories and identities. In the figure, there are different user repositories for the EIS, portal, internal consumer, and external consumer. Additionally, an enterprise user repository stores common identity information for the enterprise.

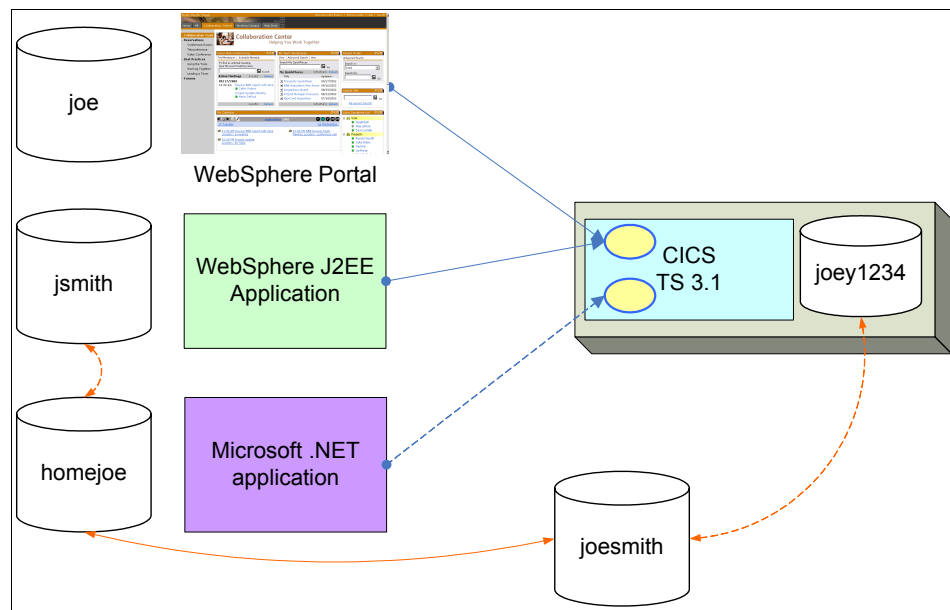


Figure 3-7 Identity environment for the SOA Foundation Service Creation Scenario

Each of these repositories have different identity and attributes for the user. Some of these attributes will be common between the repositories and a change in one must be updated in others. For example, the user may update their address in the portal user registry and this needs to be propagated to the enterprise user registry.

User repository synchronization technology is often used to do this, where a change in one repository is detected, and other repositories that need to be synchronized are updated. This is shown in the figure with a synchronization

connection between the enterprise and EIS, enterprise and portal, and portal and internal consumer repositories.

Identity provisioning

To manage the identities and attributes in each user repository effectively, an identity provisioning solution may be required, as shown in Figure 3-8. This provisioning solution can create, delete, and modify individual account information about all of the user repositories. The advantage of this approach is that a policy based provisioning solution allows only the right identity and attribute information in each of the repositories. This policy can be configured centrally, and changes are driven from this central location.

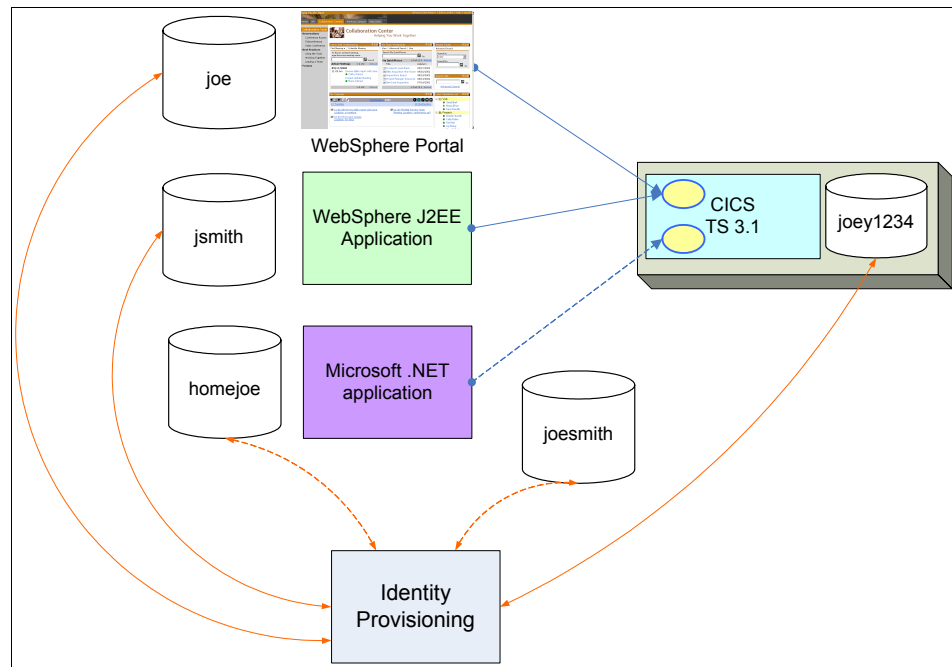


Figure 3-8 Identity Provisioning for the SOA Foundation Service Creation Scenario

One special case of provisioning is that to the external consumer user repository with user identity *homejoe*. This approach is often termed *federated provisioning*, as provisioning is occurring to another identity domain. In this case, federated provisioning standards such as Service Provisioning Markup Language (SPML), should be used to enable interoperability between different technologies.

An identity provisioning solution may also provide user self-service and password synchronization capabilities. For example, the provisioning solution might allow a user to self-enroll to a service, update their own information, and

update their own password, all without requiring intervention from an IT administrator.

Identity federation

As part of the transaction flow, secured identity information is required to flow through the environment, as shown in Figure 3-9. An important component within the Identity Service is the *trust service*.

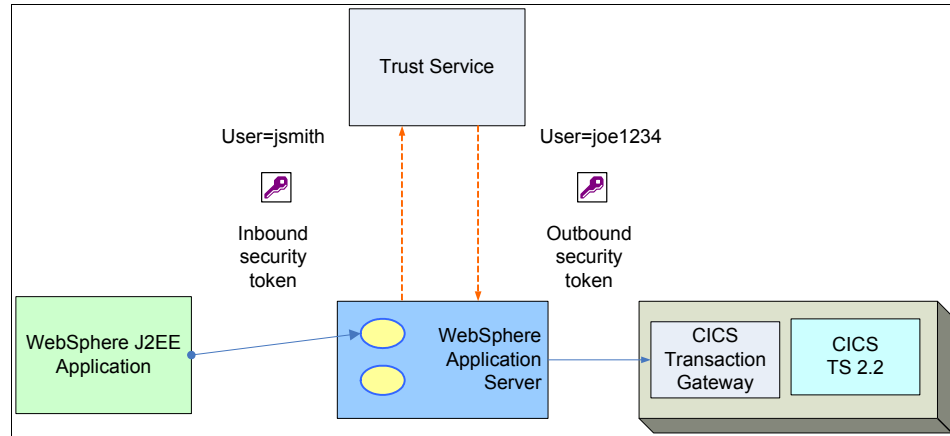


Figure 3-9 Identity Federation for the SOA Foundation Service Creation scenario in the direct architecture pattern

In Figure 3-9, the request from the internal consumer carries a *security token*. The EIS calls on the trust service to translate the incoming token from the inbound request format to one suitable for the EIS. It also may perform additional authentication, authorization, and identity mapping to translate identity and attribute information. For example, the incoming user name of *jsmith* is translated to the EIS user name of *joey1234*.

Examining Figure 3-10 on page 71, with the indirect architectural pattern, the position of the identity federation trust service is different. In this case, it is more likely to be accessed from the service components.

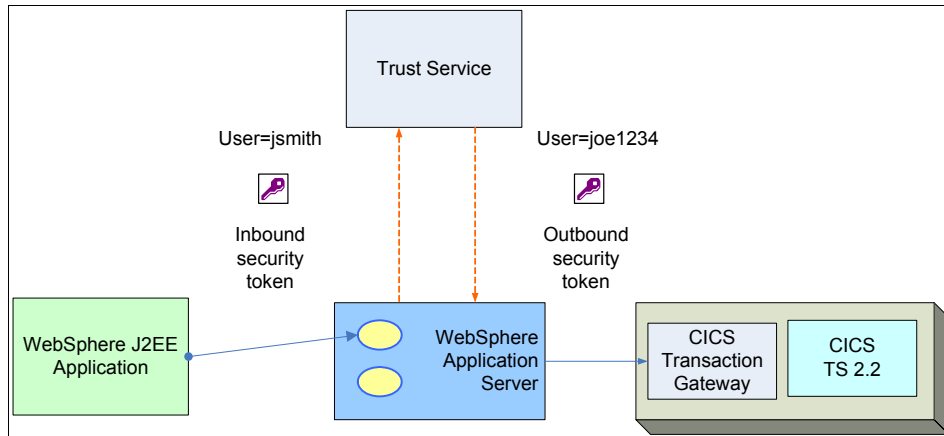


Figure 3-10 Identity Federation for the SOA Foundation Service Creation scenario in the indirect architecture pattern

Authentication Services

The Authentication Services provide capabilities to help issue and validate authentication credentials and security tokens. The Authentication Services should accommodate multiple security mechanisms, such as user name/password, Kerberos, SAML, and PKI.

Figure 3-11 on page 72 shows the application of the Authentication Services for portal users in the direct architectural pattern of the Service Creation scenario. The portal users may be requested to present authentication credentials to verify their identity to the environment. This is necessary as a pre-condition for any authorization to services. Authentication may involve a user name/password, token, or biometrics. These authentication credentials are passed to the authentication service that then verifies the identity of the user. In the figure, the user name and password are checked against the portal user registry.

Following authentication, a security token may be sent as part of the transaction flow from the portal. The trust service is called to issue the security token. The trust service is passed the authenticated user identity and returns a security token that will be passed to the EIS. The security token might again carry authentication credentials. For example, a user name token, which contains a user's name and password or user name and RACF Passticket. A binary token may carry a Kerberos ticket

(<http://www.ietf.org/html.charters/krb-wg-charter.html>). Alternatively, the security token might carry an identifier assertion only, with the receiver assuming authentication has already taken place and accepting the user identity.

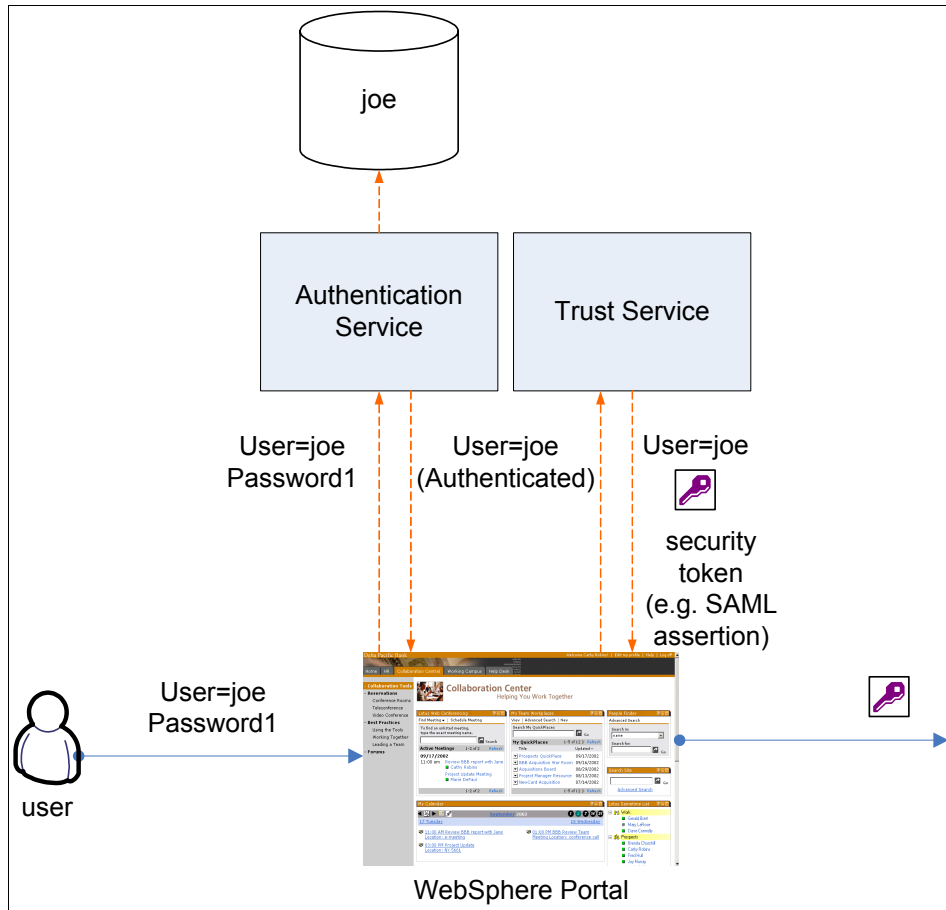


Figure 3-11 Authentication Services (user) for SOA Foundation Service Creation Scenario

The security token from the portal is sent to the EIS, as shown in Figure 3-12 on page 73. The EIS may call on the trust service to convert the incoming security token to one suitable for the EIS. This step is unnecessary if the portal has already created a security token suitable for the EIS.

If the security token carries authentication credentials, then the trust service may call on the Authentication Service to validate the request. In the figure, the Authentication Service is called from the trust service to validate the authentication credentials against the EIS user repository. If the EIS is configured to accept an identity assertion from the portal, then authentication is not required.

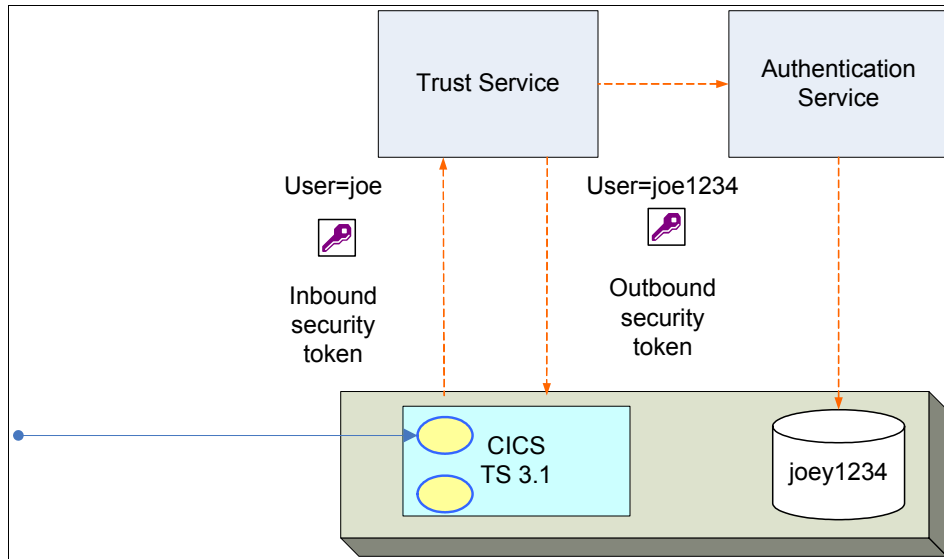


Figure 3-12 Authentication Services (token) for SOA Foundation Service Creation Scenario

Authorization and Privacy Services

Figure 3-13 on page 74 shows the application of the Authorization and Privacy Services for the Service Creation scenario. Authorization is shown at four points in the figure:

- ▶ *Service consumer:* For the portal and J2EE (internal) consumers, authorization can be implemented to control what the user can see and do. The authorization might be externalized to the Authorization Service for these consumers, or might be implemented internally within the consumers.

For the portal and J2EE application, authorization can be role based. In the J2EE security model, the authorization decision is based on *which roles can access the resource* and *can the user invoke any of these roles*. Additionally, it is common to externalize the authorization decision to an Authorization Service.

- ▶ *Service provider:* The authorization at the service provider is fairly coarse grained and is concerned with service level authorization. That is, *can the user access the operation on the service they are trying to access?* Again, the service level authorization can be externalized to the Authorization Service or can be internal to the service provider interface.
- ▶ *Application:* In most cases, there will be finer grained authorization at the application level controlling what the user might be able to do. Again, the authorization can be externalized or internal to the application. For example,

in a CICS based application, authorization rules are set up against internal security mechanisms, such as RACF.

- *Data*: There may be further authorization required at the data level, in most cases configured at the database level. This is generally not externalized to an Authorization Service.

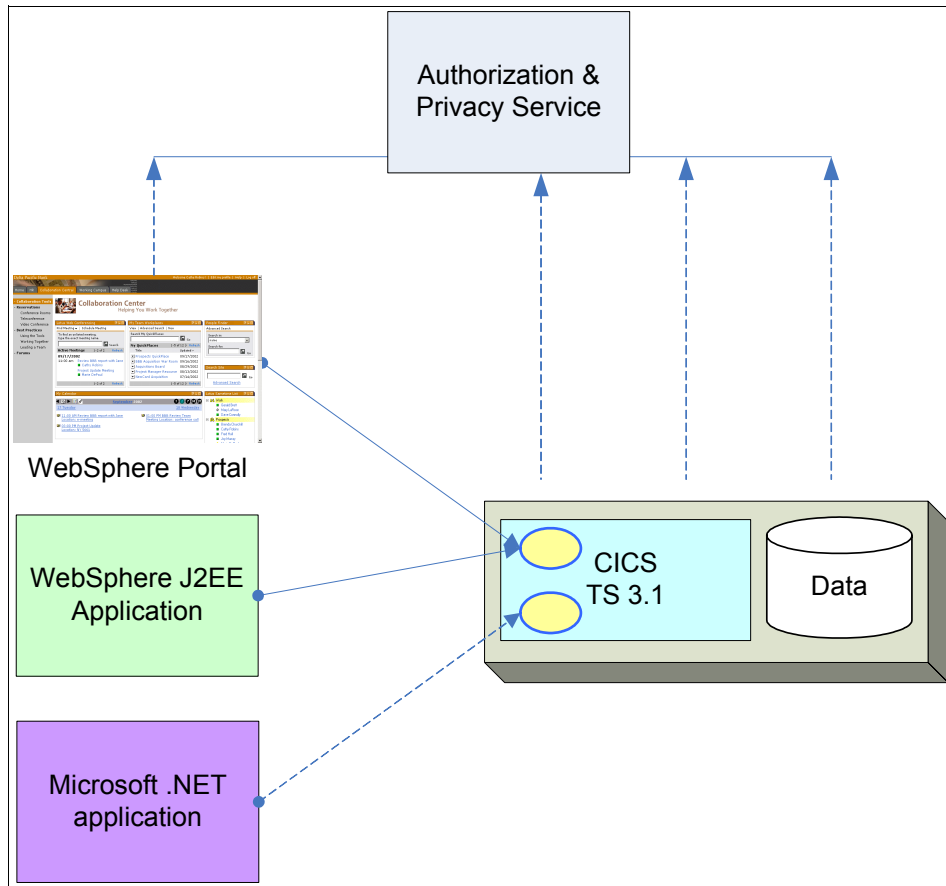


Figure 3-13 Authorization and Privacy Services for SOA Foundation Service Creation scenario

An important discussion is around the privacy aspects of authorization. Privacy authorization is controlling access to Personally Identifiable Information (PII). This type of authorization is data driven and dependent not only on the type of data, but also on user individual preferences.

Confidentiality and Integrity Services

Data protection applies to stored data. It means the different data stores need to be secured to prevent unauthorized access. Machines, folders, and files have also to be protected from external or internal threats. In this scenario, the different user registries and service provider databases in the infrastructure have to be protected, as well as the machines on which they reside.

Protecting message content from being disclosed, modified without detection, being sure of its origin, and protected against message replays are the primary concerns of a message protection service. This is usually achieved by encrypting and digitally signing a message body, a header, or any combination or parts of them.

Figure 3-14 shows the application of the data and message protection services for the direct architectural pattern of the Service Creation scenario. In the figure, data protection is shown for the CICS application data only. However, data protection in reality will be required any place there is sensitive data, including passwords, cryptographic keys, configuration files, and so on.

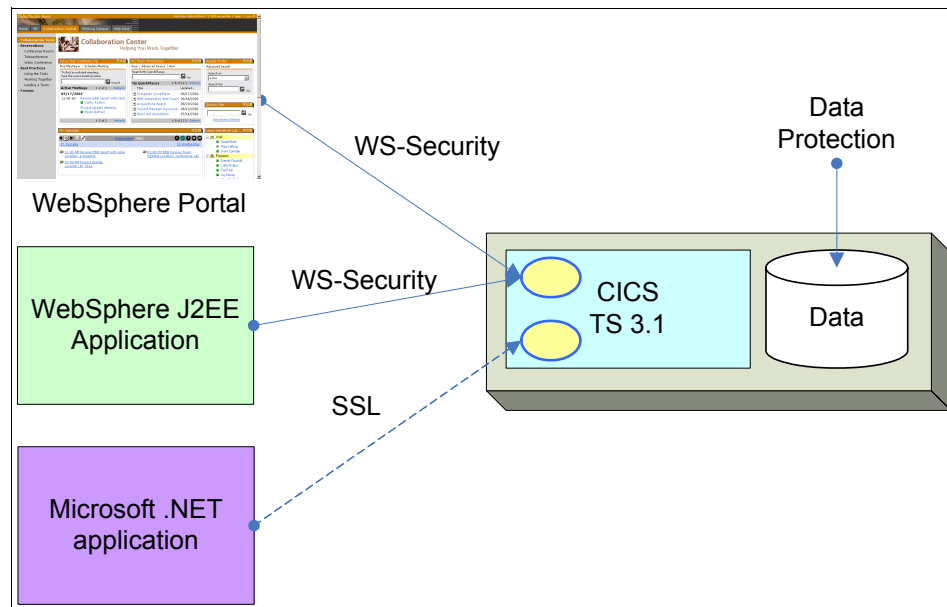


Figure 3-14 Message Protection Services for SOA Foundation Service Creation

The messages from the service consumers to the EIS are protected. Protection can take two forms:

- ▶ *Point to point protection:* Normally the whole data stream is protected at a protocol level below the message level. Secure Sockets Layer (SSL) is the most common example of a point to point protection scheme. SSL is used to secure the transport level that is at a protocol layer below the message layer, so all data that flows over the SSL connection is protected. The disadvantages of this approach is that data is unprotected in intermediate nodes and this type of protection does not allow selective protection of message content. In Figure 3-14 on page 75, the service requests from the portal are protected using SSL.
- ▶ *Message level protection:* The actual message, or some parts of it, are protected at the message level. Therefore, even if the message passes through intermediate nodes, it is still protected. In Figure 3-14 on page 75, WS-Security is used to secure these messages.

It is possible to combine the two message protection models. For example, the message may be signed using message level protection. It may then be sent across an SSL connection for confidentiality protection.

Note that in the indirect architectural pattern, messages will need to be protected both from service consumer to service components, and service components to EIS.

Audit Services

The audit logging services are in place to understand the operation of the security environment and to be sure that it is compliant with policy. To do this, an audit service provides the following:

- ▶ *Mechanisms* to submit, collect, persistently store, and report on audit data submitted as events.
- ▶ *Methods* to check compliance of the events to the individual security service policies.

Audit records can be collected centrally, or be presented centrally via one virtual view.

Table 3-1 on page 77 shows the application of the Audit Services for the Service Creation scenario. The table demonstrates that an audit event may be created and stored potentially for every identity, authentication, authorization and privacy service, and message protection call.

Table 3-1 Summary of Auditing Required for Service Creation Scenario

Security Service	Auditing Event
Audit Service	<ul style="list-style-type: none"> ▶ Identity (foundation) ▶ Identity (provisioning) ▶ Identity (federation) ▶ Authentication (consumer) ▶ Authentication (EIS) ▶ Authorization and Privacy (service consumer) ▶ Authorization and Privacy (service components) ▶ Authorization and Privacy (EIS) ▶ Authorization and Privacy (EIS data) ▶ Data protection (EIS) ▶ Message protection (service consumer to service component) ▶ (indirect case) Message protection (service component to EIS)

Non-repudiation Services

The Non-repudiation Services are in place so that there is evidence that a transaction has taken place. Let us recall the two aspects as previously described:

- ▶ Protect the recipient from a false denial by an originator that the data has been sent.
- ▶ Protect an originator against a false denial of a recipient that the data has been received.

From the point of view of the service provider in the Service Creation scenario, the important evidence to record is that the consumer has made a request to the service. This implies receiving digitally signed data from the consumer. This may have been created by the Confidentiality and Integrity Services. Additionally the consumer may have authenticated the user with the Authentication Service. A non-repudiation record can then be created at the service provider by calling on the Audit Service to record the signed data request with a time stamp to indicate when the request occurred.

From the point of view of the service consumer in the Service Creation scenario, the important evidence to record is that the service provider has responded to the request. This implies receiving digital signed response data from the provider

that may have been created by the Confidentiality and Integrity Services. A non-repudiation record can then be created at the service consumer by calling on the Audit Service to record the signed data response with a time stamp to indicate when the response was received.

Summary of IT security services

Table 3-2 shows a summary of the candidate points for applying the SOA Security Reference Model's IT security services to the Service Creation scenario. Audit is not included in this table, but is treated separately in Table 3-1 on page 77. This table is a starting point when analyzing a Service Creation scenario and could be used as a checklist when implementing a solution.

Table 3-2 Checklist of IT security services

Security Services	Application to the Service Creation Scenario
Identity Services	<ul style="list-style-type: none"> ▶ Identity foundation ▶ Identity provisioning ▶ Identity federation
Authentication Services	<ul style="list-style-type: none"> ▶ Service consumer (user or system) ▶ (Indirect case) Service components ▶ EIS
Authorization and Privacy Services	<ul style="list-style-type: none"> ▶ Service consumer ▶ (Indirect case) Service components ▶ EIS ▶ EIS data
Confidentiality and Integrity Services	<ul style="list-style-type: none"> ▶ EIS data ▶ Service consumer to service components ▶ (Indirect case) Service component to EIS
Non-repudiation Services	<ul style="list-style-type: none"> ▶ Service consumer (user or system) ▶ (Indirect case) Service components ▶ EIS

3.2.2 Security Policy Infrastructure

Security policies are derived from the business requirements and govern the behavior of the security services. In the Service Creation scenario, the security policies therefore control the identity, authentication, authorization and privacy, confidentiality and integrity, audit, and non-repudiation services.

The Security Policy Infrastructure shown in Figure 3-15 on page 79 is the building block on which to manage and deliver these policies. It should be effective and flexible, and built on open standards.

In the Service Creation scenario, several enforcement points are defined to apply security. Each of these Policy Enforcement Points (PEP) uses a Policy Decision Point (PDP) that relies on the different policies defined in the infrastructure to make the appropriate security decision.

It is important to manage these security policies so that they are correctly transformed and distributed to all the components that need to be security aware.

The Security Policy Infrastructure does not depend directly on the Architectural Pattern chosen for this scenario (direct or indirect exposure). However, the pattern chosen does affect the position of some enforcement and decision points.

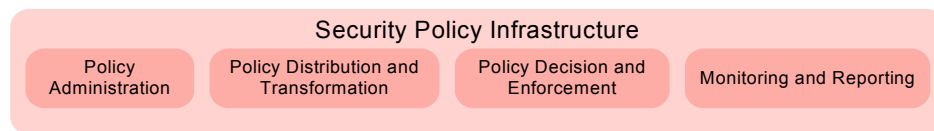


Figure 3-15 Security Policy Infrastructure from the SOA Security Reference Model

Policy administration

The policies have to be managed for all the decision points defined in the scenario. The indirect architectural pattern is used for much of the discussion in this section that describes the different policies that apply to these scenarios. As defined in the SOA Security Reference Model, *message protection policies* and *provider policies* make up these policies.

Message protection policies

In this scenario, the service provider is responsible for providing the different configuration elements to the different service consumers.

For example, an external service consumer may add digital signature and encryption to the content of the message, as the transport layer security is not considered enough security. An internal application accessing the service through the service components may rely on a security token without signature.

In terms of the security token used in the scenario, an external application may use a SAML token while an internal application uses a user name token.

In the indirect architectural pattern, the EIS component provides some policy information to the service components. This includes the appropriate information required to generate a valid Passticket (the shared key and the application identifier) as well as the endpoint information.

Provider policies

The service consumers may use their own policies. For example, an internal application may rely on a proprietary database to store its access control policies. A portal may use a database or an external access control system. The security policies defined in the infrastructure have to be mapped to these different repositories (as specified in Policy Distribution and Transformation below).

The service components need to authenticate the users based on the information they provide, as described in “Authentication Services” on page 71. It can perform some coarse-grained authorization to restrict the access to a service.

The policies can be externalized through declarative security to rely on external components or can rely on proprietary stores.

A key driver for this scenario is *reusability*. The EIS reuses most of its existing policies.

Policy distribution and transformation

The mapping of generic security policies to the local ones depend on the technical requirements of the components involved.

The service consumers need to know, for example, what kind of security tokens they have to use and the algorithms supported for ensuring message protection.

For example, the external service consumers may have to use a SAML assertion as security token as well as message level security. Algorithms supported by the service components for ensuring message level security thus need to be provided. On the other hand, the internal applications who have to use user name tokens may not have the same constraints on the message security layer.

The service components have to be configured so the authorization rules that apply to the service are correctly enforced (see “Provider policies” on page 80 for more information). If a specific access control module is used, then the policies have to be mapped to this component. The service components also need to expose all the information required for a service consumer to invoke the service.

Figure 3-16 on page 81 shows how this process can be achieved within this scenario.

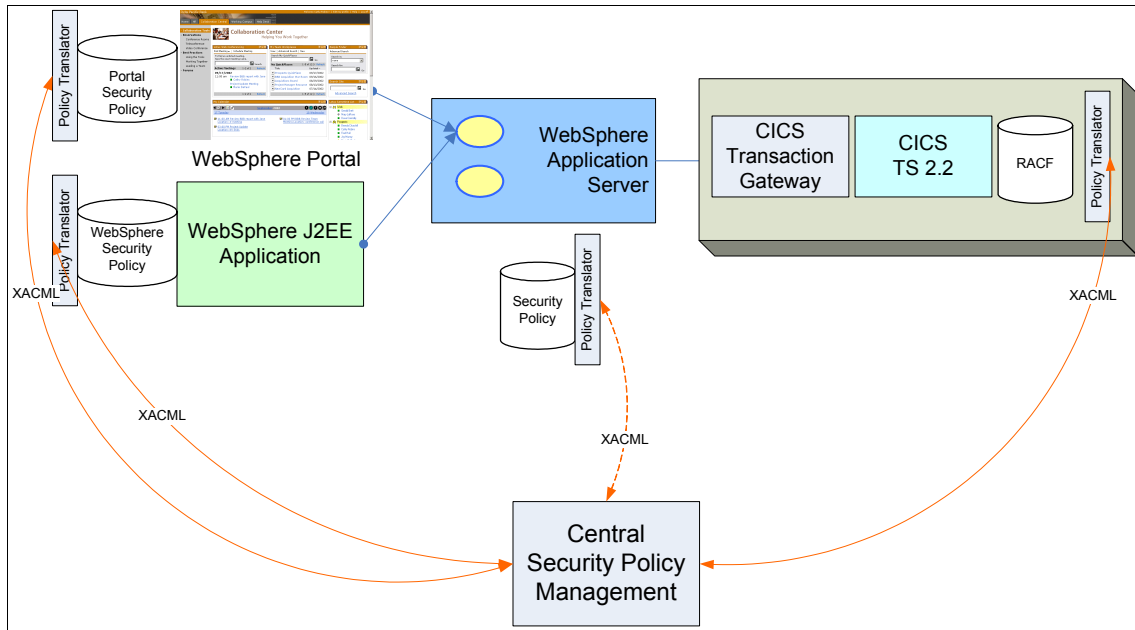


Figure 3-16 Policy Transformation and Distribution for the Service Creation scenario

For example, the authorization policies need to be pushed to the appropriate security repository used by the application server (it can be a proprietary repository or a database).

Finally, the EIS may require a mapping to the EIS specific application configuration and policies.

More information about the content of these policies are defined in the sections on “Message protection policies” on page 79 and “Provider policies” on page 80.

Policy decision and enforcement

The enforcement and decision points used in this scenario are the following:

- ▶ *The service consumer:* In the case of a user accessing the service through a portal, the portal (or a security enforcement point in front of the portal) can provide enforcement to the service access. An internal authorization database or an external authorization manager is thus used as the Policy Decision Point to provide some granularity on the access to the service. For example, a group of users may be authorized to invoke the service.

For an external service consumer, the internal authorization checks may allow only users who have federated their accounts with the service provider to access the service.

- ▶ *The service components:* In this scenario, the service components can rely on the trust service infrastructure as a Policy Decision Point. This can include the following activities:
 - Control message protection for external service providers.
 - Validate incoming a security token and a user name / password against a registry for internal consumers and a SAML assertion for external consumers.
 - Authorize users to access the service based on authorization policies.
 - Audit the incoming requests and the users accessing the service.

The security policy needs to be externalized outside the service components themselves (in artifacts such as *deployment descriptors* for example) so they can be changed and updated iteratively.

Note: More information about the approaches of *infrastructure-managed* and *application-managed* security is described in the “SOA Programming model for implementing Web services, Part 7: Securing service-oriented applications” available at <http://www.ibm.com/developerworks/webservices/library/ws-soa-progmodel7/index.html>

Even when using programmatic security, the same policy decision points can be reused.

- ▶ *The EIS:* The EIS relies on its own Policy Decision Point. The EIS validates the RACF Passticket token provided and maps it to a local identity. Finer-grained authorization based on the system policies can also be applied.

Monitoring and reporting

Based on the information provided within the audit infrastructure and the policy distributed to the decision points, reports can be generated to contribute to the compliance process. Activity reports are also made available for auditing.

3.2.3 Business Security Services

Figure 3-17 on page 83 shows the security management components of the IBM SOA Security Reference Model. The following sections discuss each of the components of security management and how they apply to the scenario.

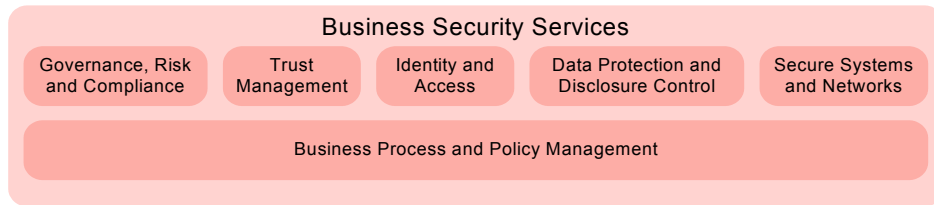


Figure 3-17 Business Security Services from the SOA Security Reference Model

Governance, risk, and compliance

An effective security governance framework involves establishing chains of responsibility, authority, and communication to empower people to effectively control the system. Governance is very important for the security services, as managing the security policy and implementation is vital to the integrity of the environment.

Governance in the Service Creation scenario involves monitoring compliance of the security services with the security policies, monitoring compliance with governance structures in place, and monitoring the overall security effectiveness of the environment.

Security compliance management measures the performance of the security implementation relative to the measures defined by the security policy. These can be realized based on reporting on system behavior using audit information and comparing that behavior to configured policies in systems. When these are viewed in the context of business defined policies, it can provide an over-arching view of where a business stands in implementation and enforcement of intended policies.

The risk management process can result in some compliance objectives that may include, for example:

- ▶ A systems administrator may need to know when accounts are created on systems that are against policy, for example, if RACF user group memberships are changed against policy.
- ▶ Passwords for users to systems may need to be updated periodically.

Figure 3-18 shows how security compliance can be managed in the Service Creation scenario. Using the Policy Management and Auditing tools available, an auditor could check that the system is behaving as expected based on policy.

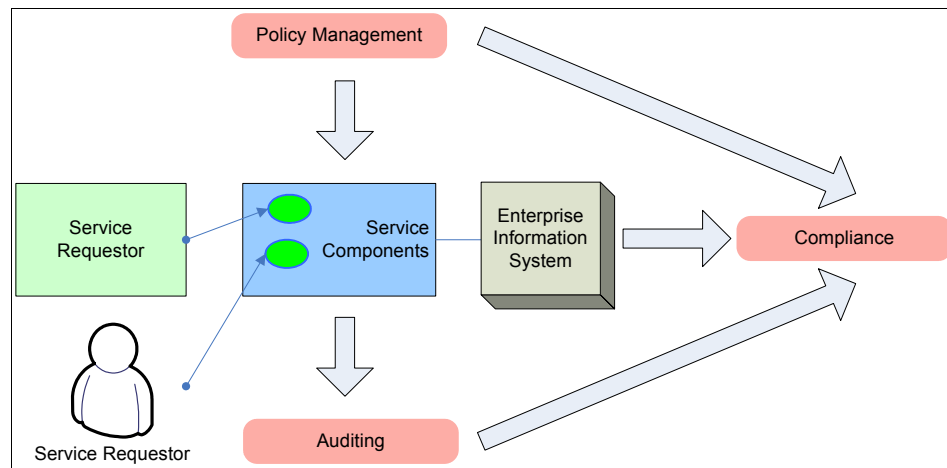


Figure 3-18 Security compliance requires examination of policy and audit information

Trust management

From a business viewpoint, trust management includes the liability and legal aspects around the services. It also includes protection messages that the service provider can implement for sensitive data.

At a technology level, trust management may include:

- ▶ The protocols for the service consumer to contact the service provider. For example, this may require a SOAP message carried on HTTPS.
- ▶ The security token and its contents that need to be included in a WS-Security message. For example, a SAML 1.0 security assertion carrying role based information is required.

Identity and access

Managing the security regarding identity and access involves defining the appropriate security policies regarding the identity and access management infrastructure.

Identity management

As identified in “Identity Services” on page 68, several user repositories are used and the user identities have to be defined in these repositories with the appropriate access rights in these systems. Synchronization between the different repositories ensures the information is updated when it changes.

A provisioning policy is, for example, defined to automatically create the user accounts in the portal repository (an LDAP directory or a database, with an account identifier *joe* for Joe Smith), in the enterprise repository (account identifier *joesmith* for Joe Smith) and in the EIS system (a RACF account with *joey1234* as the identifier for the same person Joe Smith). This provisioning policy can be extended to cross the company boundaries so the user is also created in the registry used for the external service consumer (account identifier *homejoe* for Joe Smith).

This provisioning policy may include several workflow activities, for example, getting user management approvals or system administrator approval for RACF account creation.

As part of these provisioning policies, identifier and password policies have to be taken into account. Identifier policies define how the different attributes for the different accounts are created based on the user identity information and the company security rules.

In this example, an identifier policy may be defined to create the accounts on a system using the first letter of the first name and the letters of the family name for a user (for example, *jsmith* for Joey Smith). This can also apply to other attributes, such as an e-mail address.

Password policies can be used to enforce the way passwords for the different user accounts are created and managed. For example, it can be decided to define a policy requiring a minimum length, the inclusion of numeric and special characters, and an expiration date so that the user needs to change his password every three months. This enforces security, as password weakness is known as a common risk for the systems.

Federation policies are another foundation layer of the scenario. They allow the validation, mapping, and exchange of the different security tokens that are used between the domains or systems.

On the service consumer the users may use a user name and password to authenticate to their portal or a strong authentication (for example, *homejoe*, *jsmith*, or *joe*).

A SAML assertion may be required at the service components level to authenticate a user accessing the service through an external consumer while a user name token can be enough for an internal application. The user name token provided may be different from the one used to authenticate to the local portal (for example, *joesmith*).

This identity then needs to be mapped to the appropriate user identifier in RACF (*joey1234*) so the RACF Passticket can be generated. Finally, on the EIS, the Passticket is validated and mapped to the local account identity.

In the case of the direct exposure scenario, the internal service consumers have to use the trust service to exchange the identity information they have regarding the user to a valid RACF Passticket. For example, the local identity *jsmith* or *joe* is exchanged to a Passticket token generated for user *joey1234*. The EIS then validates the Passticket.

The trust service needs to be configured so the security tokens are correctly mapped to ones suitable for the receiving entities in this scenario.

Access management

The definition of the access management policies covers the Authentication Services as well as the Authorization and Privacy Services. This requires the definition within the policy infrastructure of the appropriate access control policies entitling access to authorized users. These definitions can change depending on the component performing the security enforcement, as they may not use the same access control policy format.

The service components may define a first set of authentication and authorization requirements to prevent unauthenticated users to access the service. Then finer grained authorization can be done through J2EE role-based security.

Finally, on the EIS system, the CICS security policies can be defined to authorize only account owners to view their account information.

Data protection and disclosure control

Data protection management uses the services described in “Confidentiality and Integrity Services” on page 75. Data protection management identifies the resources that need protection and the controls required on those resources. For example:

- ▶ Where possible, cryptographic key stores should be located on dedicated hardware for secure storage.
- ▶ Where cryptographic key stores on the file system have to be used, they should be protected with operating file system permissions, and stored on an encrypted file system.
- ▶ An operating system hardening solution should be used to protect service provider machines, and best practices be employed for identifying the files to be protected, based on the type of application server used.

Disclosure control uses the Authorization and Privacy services introduced in 3.2.1, “IT Security Services” on page 67. An organization may publish its privacy policy for users to review prior to subscribing to services that it provides, allowing them to make an informed decision before opting-in to using these services. A reporting mechanism should also be available to provide data on disclosure control to the compliance function.

User consent is almost always a component of a disclosure control implementation. For example, a user’s consent may be required before a user’s accounts are federated. Another aspect of disclosure control might be that a user has access to a portlet that shows what personal information the organization is retaining about them.

Secure systems and networks

Managing the security of the deployment environment where business solutions can be deployed and hosted is important. There are a set of tools that help protect infrastructure servers, systems, and networking resources from security threats. Solutions in this category provide protection against viruses, hackers, and misuse by internal users.

As availability relates to security, the performance and availability indicators that need to be monitored have to be defined. For example, the unsuccessful login attempts and some response times of the system can be monitored.

Note: More information about monitoring activities within SOA are available in the IBM Redbook *IBM Tivoli Composite Application Manager V6.0 Family: Installation, Configuration and Basic Usage*, SG24-7151.

Business Process and Policy Management

In the Service Creation scenario, some aspects of Business Process and Policy Management include:

- ▶ Processes around periodic assessment of services to ensure they comply with appropriate policies.
- ▶ Ensuring that no services are unknown to the governance board.
- ▶ Processes around managing trust relationships with external service consumers.
- ▶ Processes around managing the life cycle of identities in the service provider. For example, this includes how new users get the RACF accounts used by the CICS application.
- ▶ Processes around ensuring that the service provider manages business sensitive data appropriately.

3.3 Summary

The application of the IBM SOA Security Reference Model to the SOA Foundation Service Creation scenario has been described in this chapter. There are many different security issues to contend with, and these must be applied across the different components of the scenario. This chapter is therefore usable as a checklist in determining where security can be applied to a real customer environment.



IBM SOA Foundation Service Connectivity scenario

This chapter describes the application of the SOA Security Reference Model, as defined in Chapter 2, “Architecture and technology foundation” on page 17, to the SOA Foundation Service Connectivity scenario. This scenario is the second of the IBM SOA Foundation scenarios. The chapter highlights how each component of the IBM SOA Security Reference Model can be applied to the scenario.

4.1 Scenario overview

In the SOA Foundation Service Creation scenario given in Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61, there is a direct connection between the service consumer and service provider. In that scenario, service consumer and service provider can be considered tightly coupled. The service consumer needs to know the location of the service provider, and needs to conform to the exact service specification of the service provider. That is, the communication protocols used to invoke services and the data formats used to exchange input and output data, have to be agreed upon and followed. Any change in the service provider service specification will result in a change required in the service consumer.

The decoupling of service consumer and service provider provides a number of benefits. It allows for the substitution of one service provider for another. For example, another provider may offer the same service for lower cost or with higher quality of service. The service provider can be changed without the consumer being aware of the change or without the need to alter the architecture to support the substitution.

Figure 4-1 on page 91 shows the SOA Foundation Service Connectivity scenario. The elements are similar to those of the Service Creation scenario; however, there is a new component: the *Enterprise Service Bus* (ESB).

The ESB is a key enabler for SOA and represents a broad range of capabilities. At a minimum, the ESB should have the following capabilities:

- ▶ *Routing*: Ensure that any request a consumer initiates is sent to the correct provider.
- ▶ *Addressing*: Addressing complements routing to provide location transparency and support service substitution. Service addresses are transparent to the service consumer.
- ▶ *Messaging styles*: Should support a variety of messaging styles. The most common are request/response, fire and forget, events, publish/subscribe, and synchronous and asynchronous messaging.
- ▶ *Transport protocols*: Support protocol transformation between consumer and provider. The most common of these is HTTP, but other protocols are used depending on the method used to expose the service.
- ▶ *Service interface definition*: Should have a formal service definition, such as WSDL.
- ▶ *Service messaging model*: Should support a model such as SOAP or XML.

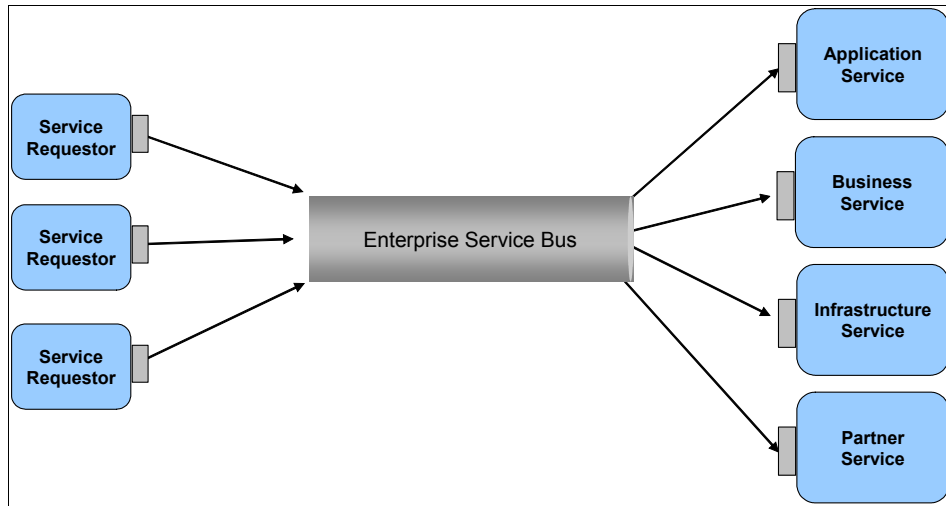


Figure 4-1 Service Connectivity Scenario

While there are multiple products (like DataPower®, WebSphere ESB, WebSphere Message Broker, and so on) that provide the functionality of an ESB, this IBM Redbook uses WebSphere ESB as the representative ESB. The decision to use which ESB depends upon the environment and business requirements.

Figure 4-2 gives an example to illustrate a real product implementation. In this case, the ESB is WebSphere Enterprise Service Bus (WebSphere ESB) and DataPower XS40 security appliance. The portal is WebSphere Portal, with its portlets generating Web service requests. The internal and external service consumer applications are WebSphere J2EE and Microsoft .NET applications, respectively. The internal service provider is CICS Transaction Services (TS) 3.1, exposing services directly.

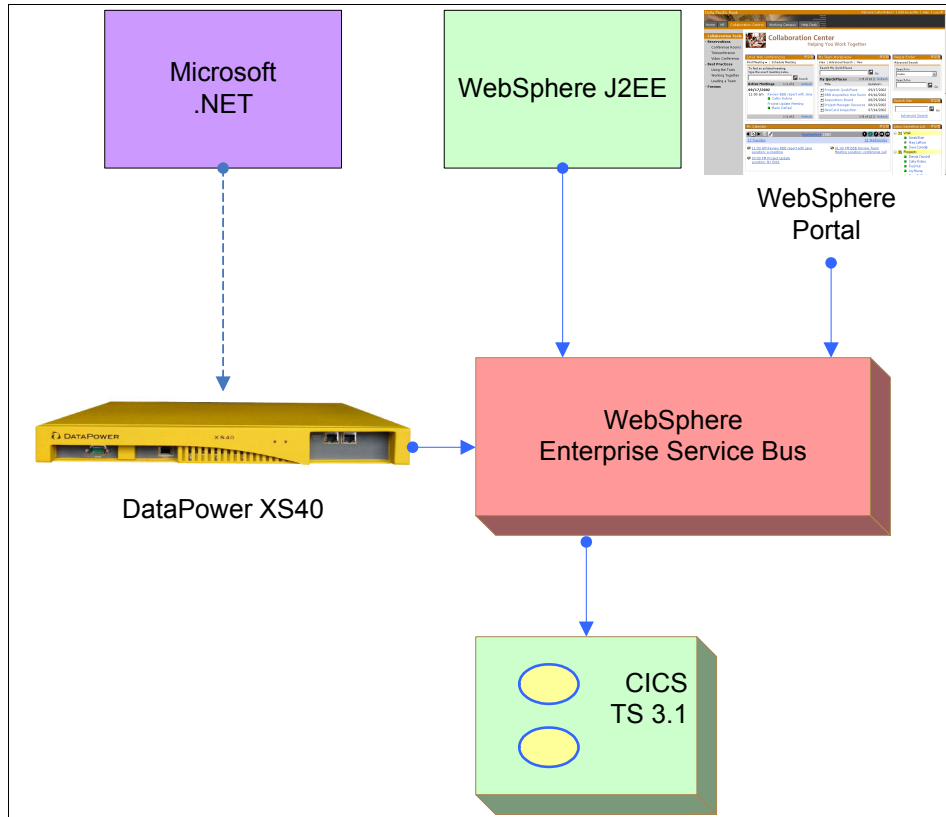


Figure 4-2 Service Connectivity Scenario - example product mappings

As shown in Figure 4-3, all of the functions of the ESB are hosted within components called *mediations*. Mediation modules contain imports, exports, and mediation logic. Mediation modules perform the message protocol transformation, content-based message routing, message format transformation, and message augmentation.

Mediation modules may use other mediation modules to perform a complete message mediation. This allows for a more flexible service assembly and developers may re-use mediation modules across several mediations. Figure 4-3 shows the use of two mediation modules between service consumer and service provider.

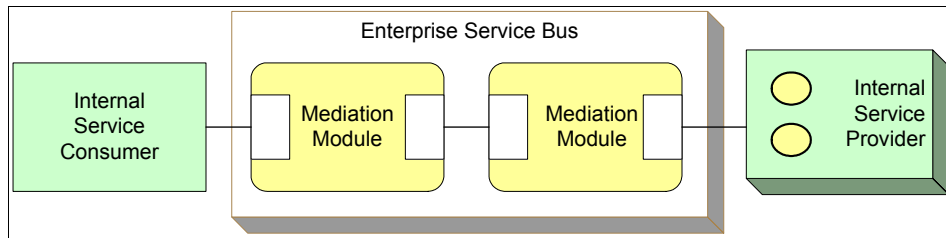


Figure 4-3 Enterprise Service Bus mediation modules

4.2 Applying the IBM SOA Security Reference Model

Figure 4-4 shows the IBM SOA Security Reference Model. There are three main components to the reference model: IT Security Services, Security Policy Infrastructure, and Business Security Services. A more detailed description of each of the components is included in Chapter 2, “Architecture and technology foundation” on page 17.

Similar to Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61, a key point to understand when reading this chapter is that it describes how each of the SOA Security Reference Model components *could* be applied to the Service Connectivity scenario. The decision about which components *should* be applied for a particular implementation of the Service Connectivity scenario depends on the relevant business requirements.

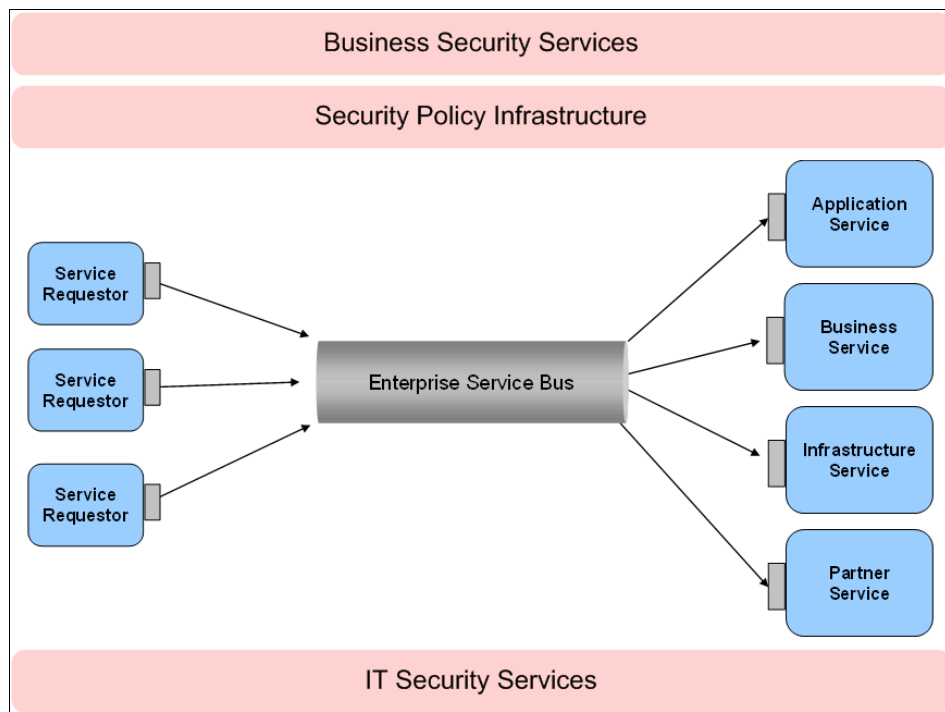


Figure 4-4 Applying the SOA Security Reference Model

4.2.1 IT Security Services

The use of common IT Security Services enables a consistent security implementation. It also minimizes development and deployment costs for implementing these services. IT security services can be used by different components in the SOA environment, such as gateways, proxy servers, application servers, data servers, and operating systems.

Figure 4-5 shows the list of IT security services from the IBM SOA Security Reference Model. We will discuss each of these security services in turn, and show how they can be used within the SOA Foundation Service Connectivity scenario.

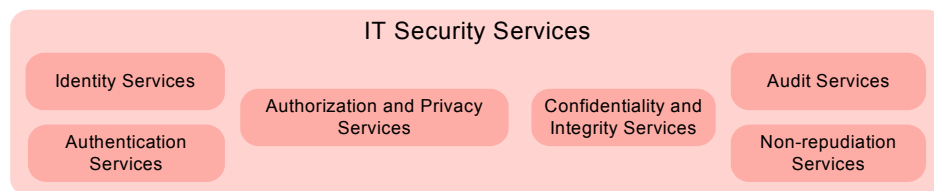


Figure 4-5 IT Security Services from the SOA Security Reference Model

Identity Services

In the Service Creation scenario section about “Identity Services” on page 68, the fundamental need for identity services was discussed. That section reviewed three particular identity services: identity foundation, identity provisioning, and identity federation. All of these identity services are required for the Services Connectivity scenario as well.

Identity provisioning

To enable a policy based approach to managing the identities across the user repositories, an identity provisioning solution can be implemented. As shown in Figure 4-7, a central provisioning service creates, modifies, and deletes identity information across the user repositories. The advantage of this approach is that it allows the central definition of provisioning policies and consistent implementation. A special case of provisioning is the federated provisioning to the external service consumer repository. In this case, open standards based provisioning technology, such as WS-Provisioning or SPML, may be required.

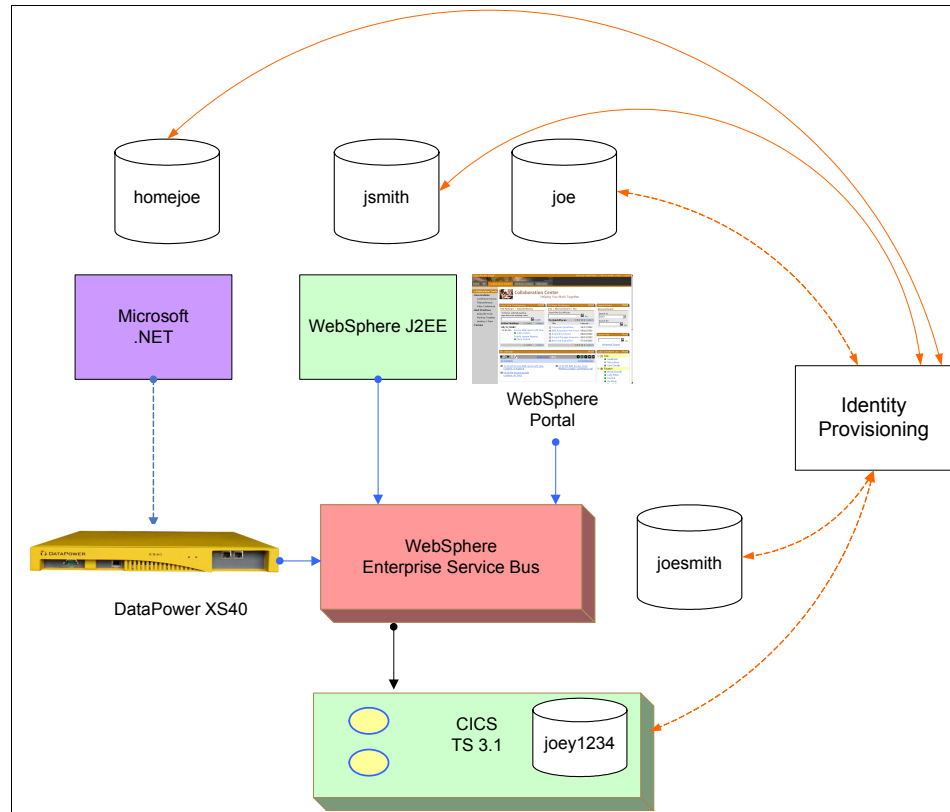


Figure 4-7 Identity Provisioning for the SOA Foundation Service Connectivity Scenario

Identity federation

As part of the transaction flow, secure identity information is required to flow through the system. Take the case of a request coming in from the external service provider. As shown in Figure 4-8, the DataPower XS40 provides a Web services gateway to the incoming requests. In the figure, the request arrives carrying a SAML security token, which carries identity information for the user *homejoe*. The point of contact calls into the trust service to exchange the security token format to one supported by the enterprise. In this case, the SAML security token is exchanged for a user name token. Additionally, the trust service can map the incoming identity information to one suitable for the enterprise. In the figure, the *homejoe* identity is mapped to the enterprise identity of *joesmith*.

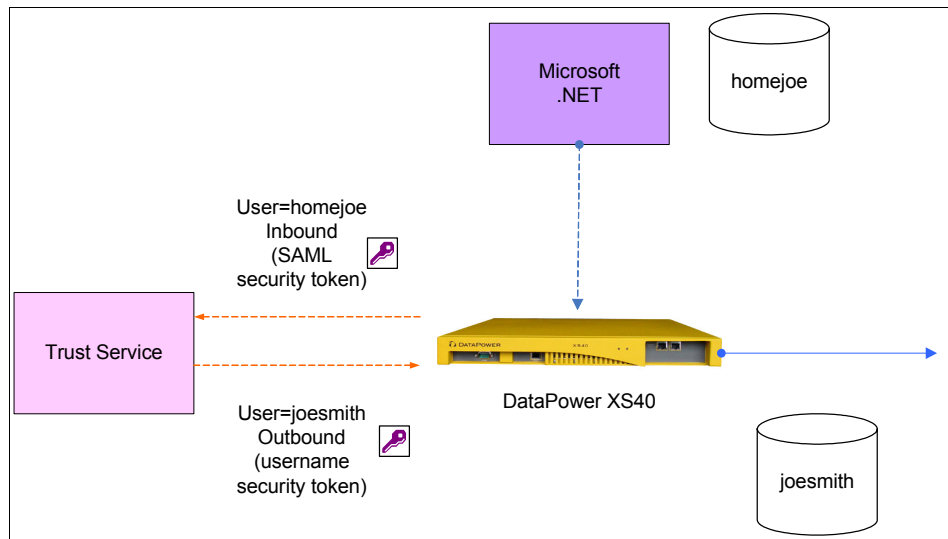


Figure 4-8 Identity federation at the Web Services Gateway

In a similar way, identity federation may be required at the ESB layer. In Figure 4-9 on page 99, the request from the gateway carries a user name security token. One of the ESB mediation modules calls on the trust service to translate the incoming token to one suitable for the EIS. In this case, it is creating a security token that carries the RACF Passticket information. It also may perform identity mapping to translate identity and attribute information. For example, the incoming user name of *joesmith* is translated to the EIS user name of *joey1234*.

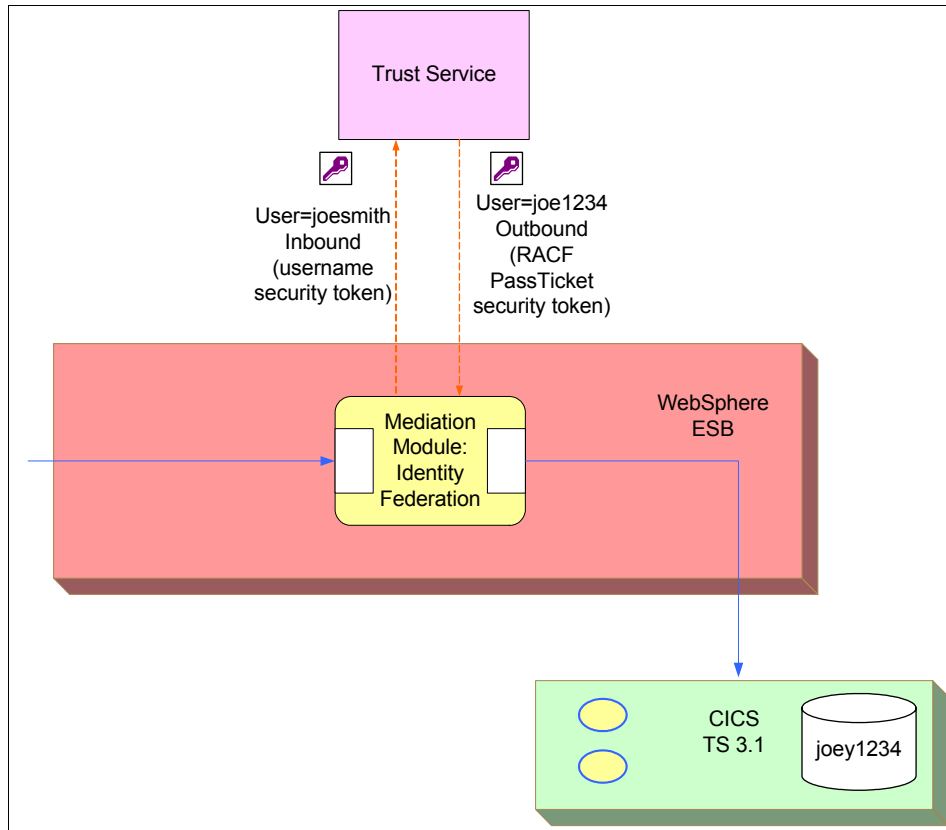


Figure 4-9 Identity Federation at the ESB

Authentication Services

The Service Creation scenario discussion in “Authentication Services” on page 71 shows the application of the Authentication Services for portal users. The application of authentication for portal users in this scenario is the same.

The portal users may be requested to present authentication credentials to verify their identity to the environment. This is necessary as a pre-condition for any authorization to services. Authentication may involve a user name/password, hardware token, or biometrics. These authentication credentials are passed to the Authentication Service that then verifies the identity of the user. In Figure 4-10, the user name and password are checked against the portal user registry.

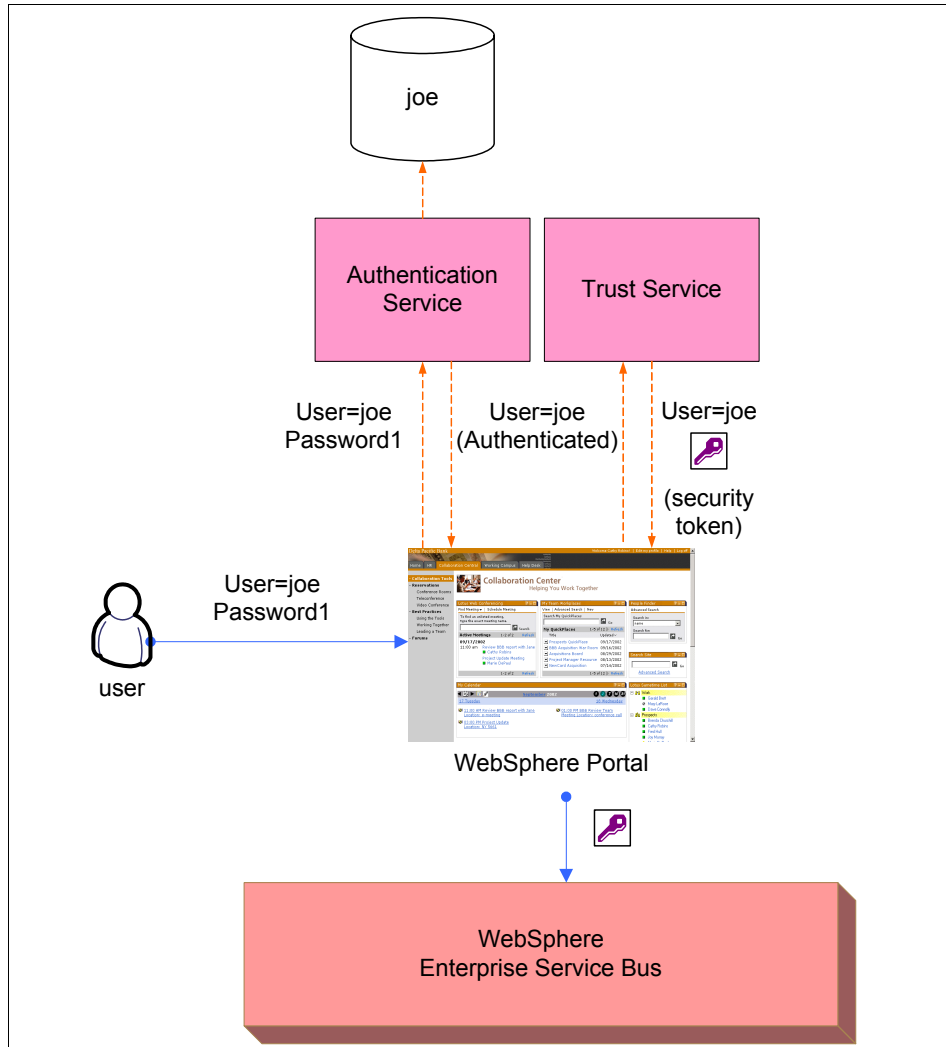


Figure 4-10 Authentication Service for the SOA Foundation Service Connectivity Scenario - Portal

The other authentication takes place at the EIS, as shown in Figure 4-11. In this case, the request from the ESB contains the security token appropriate to the EIS and the EIS could authenticate the request directly. In the figure, the EIS calls out to the Authentication Service. In many cases, the EIS may authenticate to its own user repository directly.

Note: This is a different approach as compared to the Service Creation scenario, where EIS authentication relied on a separate call to the trust service to create the token suitable for it. That example showed calling the trust service to get the RACF Passticket. The advantage of using the ESB is that it can perform identity federation for a service provider, before making any request to the service provider.

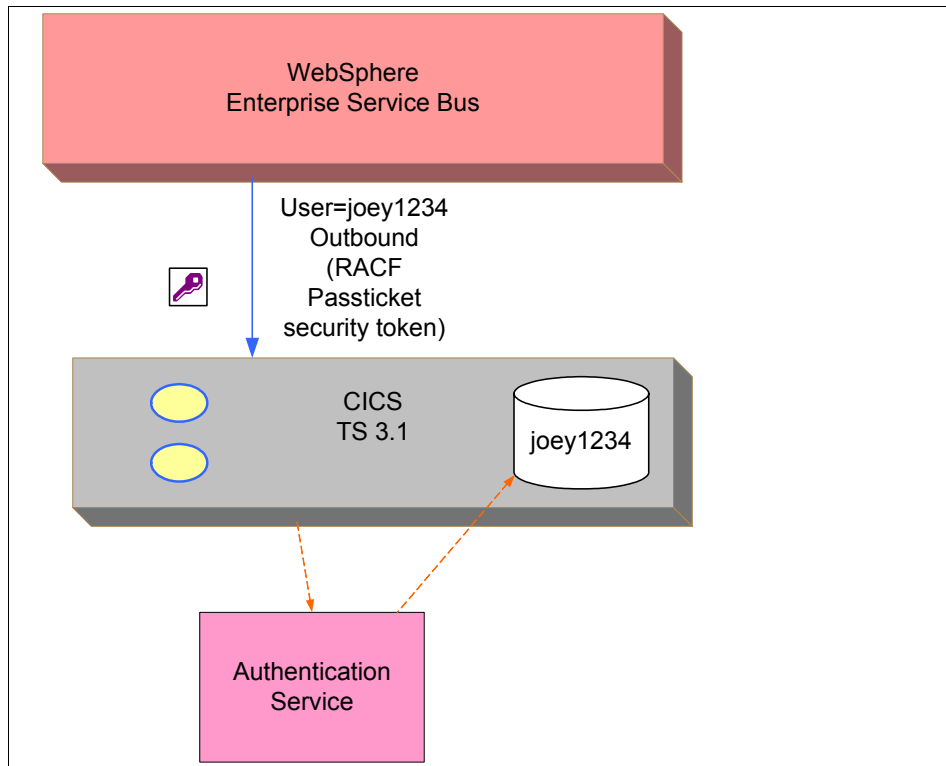


Figure 4-11 Authentication Service for the SOA Foundation Service Connectivity Scenario - EIS

Authorization and Privacy Services

The first application of the Authorization and Privacy Services is shown in Figure 4-12. Requests that come in from the external service consumer must be authorized before being granted access into the enterprise. This is an application of the normal *edge of network* control applied with firewalls, Web reverse proxies, and similar devices. In this case, the gateway should call out to the Authorization and Privacy Services to ensure incoming requests are authorized. Any requests that are not authorized should be rejected.

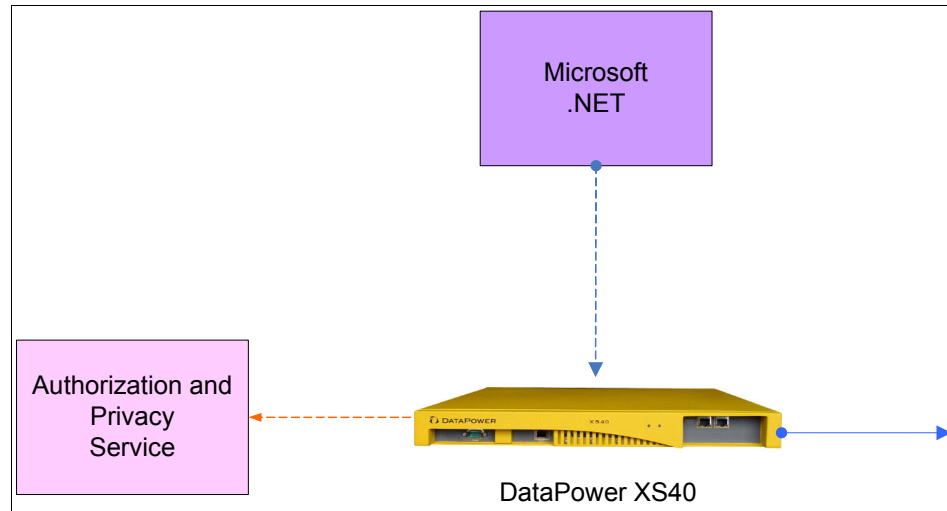


Figure 4-12 Authorization and Privacy Service for the SOA Foundation Service Connectivity Scenario - Web Services Gateway

There are four more Authorization and Privacy Services aspects within the scenario. These are depicted in Figure 4-13 on page 103:

- ▶ *Service consumer:* For the portal and WebSphere J2EE internal service consumer, the authorization can be implemented to control what the user can see and do. The authorization can be externalized to the security service, or alternatively implemented internally.
- ▶ *Enterprise Service Bus:* As shown on Figure 4-13 on page 103, the mediation module on the ESB can call out to the Authorization And Privacy Service to implement service level authorization. This level of authorization controls who can call into a service operation.

Note: This is a different approach compared to the Service Creation scenario, where service level authorization had to be implemented at the service provider. The advantage of using the ESB is that it can perform authorization for a service provider before making any request to the service provider.

- ▶ *Service application:* In most cases, there will be finer grained authorization at the application level, controlling what the user might be able to do. Similar to the Service Creation scenario, the authorization can be externalized to the service, or implemented with the application, for example, the CICS based application provides authorization using RACF.
- ▶ *Service data:* There may be further authorization at the data level, most commonly implemented at the database.

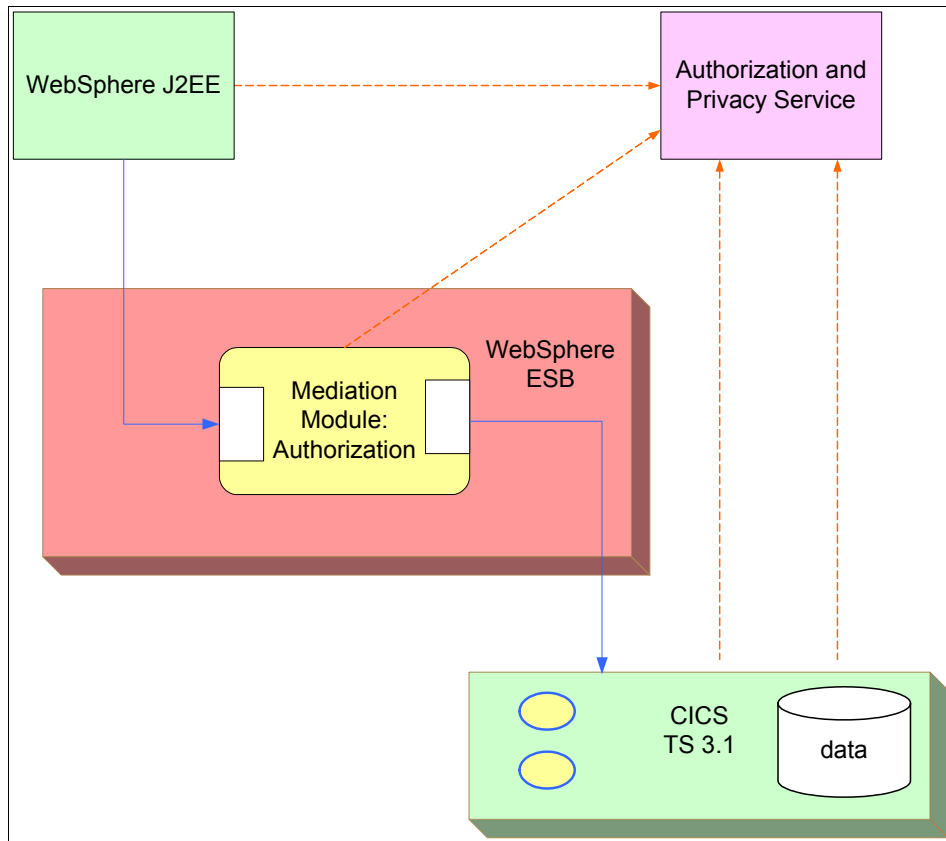


Figure 4-13 Authorization and Privacy Service for the SOA Foundation Service Connectivity Scenario

Confidentiality and Integrity Services

Data protection applies to stored data. It means the different data stores need to be secured to prevent unauthorized access. Machines, folders, and files have to be protected from external or internal threats as well. In this scenario, the different user registries and service provider databases in the infrastructure have to be protected, as well as the machines on which they reside.

In “Confidentiality and Integrity Services” on page 75 of the Service Creation scenario, we described the different types of message protection:

- ▶ *Point to point protection:* Normally, the whole data stream is protected at a protocol level below the message level. Secure Sockets Layer (SSL) is the most common example of a point to point protection scheme.
- ▶ *Message level protection:* The actual message, or some parts of it, are protected at the message level. This may involve use of digital signatures to verify the message integrity or encryption to guarantee message confidentiality.

Figure 4-14 on page 105 shows the application of the message protection services for the Service Connectivity scenario. In the figure, WS-Security is used to provide protection of messages from the external service consumer to the Web services gateway, internal service consumer to ESB, and the gateway to ESB.

There is a requirement for the ESB and Web services gateway to provide support for common message protection standards, such as SSL and WS-Security.

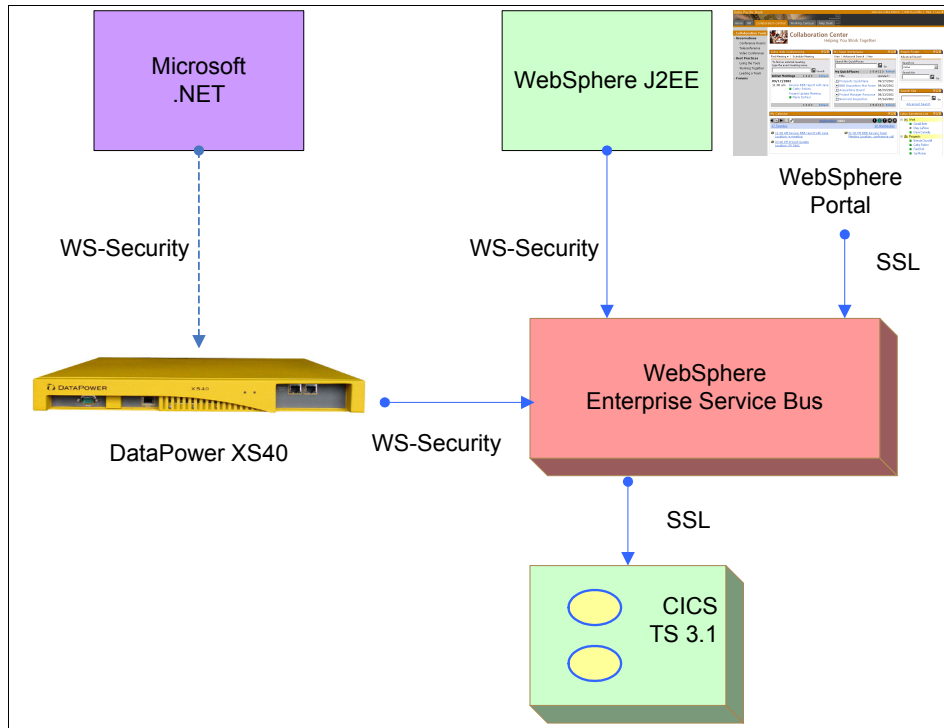


Figure 4-14 Message Protection Service for the SOA Foundation Service Connectivity Scenario

Audit Services

The audit and logging services are in place to understand the operation of the security environment and to be sure that it is compliant with policy. To do this, an audit service has to provide the following:

- ▶ Mechanisms to submit, collect, persistently store, and report on audit data submitted as events.
- ▶ Methods to check compliance of the events to the individual security service policies.

Table 4-1 shows the application of the audit services for the Service Connectivity scenario. The table is demonstrating that potentially for every Identity, Authentication, Authorization and Privacy Service, and data and message protection call, which an audit event may be created and stored.

Table 4-1 Audit Services for the Service Connectivity scenario

Security Service	Auditing Event
Audit Service	<ul style="list-style-type: none"> ▶ Identity (foundation) ▶ Identity (provisioning) ▶ Identity (federation) ▶ Authentication (service consumer) ▶ Authentication (EIS) ▶ Authorization and Privacy (service consumer) ▶ Authorization and Privacy (Gateway & ESB) ▶ Authorization and Privacy (EIS) ▶ Authorization and Privacy (EIS data) ▶ Message protection (Gateway) ▶ Message protection (Gateway to ESB) ▶ Message protection (ESB to service component) ▶ Data protection (EIS) ▶ (indirect case) Message protection (service component to EIS)

Non-repudiation Services

The Non-repudiation Services are in place so that there is evidence that a transaction has taken place. There are two aspects, as previously described:

- ▶ Protect the recipient from the false denial by an originator that the data has been sent.
- ▶ Protect an originator against the false denial of a recipient that the data has been received.

From the point of view of the service provider in the Service Connectivity scenario, the ESB first receives the initial service request, and accesses service providers to fulfill those requests. In this case, the ESB may invoke the Non-repudiation Services, based on the incoming request. The service providers accessed by the ESB may access the Non-repudiation Services to record the transaction coming from the ESB. A combination of multiple non-repudiation records may be required to associate an incoming request with the events that occurred on a set of service providers.

From the point of view of the service consumer in the Service Connectivity scenario, the important evidence to record is that the ESB has responded to the request. Unlike the Service Creation scenario, access to the service providers is not direct, so evidence for the non-repudiation record is likely to be based upon the secure channel between consumer and ESB. A non-repudiation record can then be created at the service consumer by calling on the audit service, to record the signed data response with a time stamp to indicate when the response was received.

Summary of security services

Table 4-2 shows a summary of the candidate points for applying the IBM SOA Security Reference Model's IT Security Services to the Service Connectivity scenario. An audit is not included in the table, but treated separately in Table 4-1 on page 106. This table is a starting point when analyzing a Service Connectivity scenario and could be used as a checklist when implementing a solution.

Table 4-2 A checklist of IT security services from the IBM SOA Security Reference Model that can be applied to the Service Connectivity Scenario

Security services	Application of the Service Connectivity Scenario
Identity Services	<ul style="list-style-type: none"> ▶ Identity foundation ▶ Identity provisioning ▶ Identity federation
Authentication Services	<ul style="list-style-type: none"> ▶ Service consumer (user or system) ▶ EIS
Authorization and Privacy Services	<ul style="list-style-type: none"> ▶ Service consumer ▶ Gateway ▶ ESB ▶ EIS ▶ EIS data
Confidentiality and Integrity Services	<ul style="list-style-type: none"> ▶ Service consumer to Gateway ▶ Gateway to ESB ▶ ESB to service components ▶ (Indirect case) Service component to EIS ▶ EIS data
Non-repudiation Services	<ul style="list-style-type: none"> ▶ Service consumer ▶ ESB ▶ Service provider

4.2.2 Security Policy Infrastructure

Security Policy Infrastructure in the context of SOA is the means by which processes and services express the conditions and manage the behavior of the underlying infrastructure, in order to secure access to information, information availability and retention, and audit. Policy management includes authoring business policies that get refined to service specific policies, such as security, performance indicators and metrics, and trust policies.

Figure 4-15 shows the components of Security Policy Infrastructure from the IBM SOA Security Reference Model. We will discuss each of these in turn, and show how they can be used within the SOA Foundation Service Connectivity scenario.

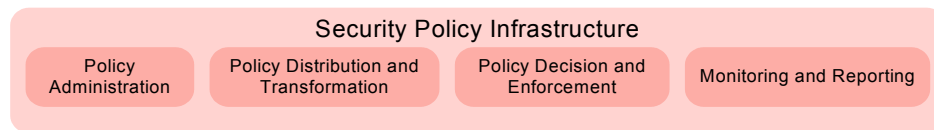


Figure 4-15 Security Policy Infrastructure from the SOA Security Reference Model

Policy administration

Policy management deals with the life cycle of security policies. It can be decomposed into *message protection policies* and *provider policies*. The policies expressed here may be expressed in an abstract, standard format, rather than specific to any components of the architecture. Policies expressed in this higher-level form are usually easier to tie back to the business perspective of the policy.

Message protection policies

Message protection policies specify the mechanisms, rules, and constraints on how a service consumer will use a service provider. In this scenario, where the ESB is an intermediary, service consumers and service providers interact with the ESB and not each other, so the message protection policies describe the interactions between:

- ▶ Service consumer and ESB
- ▶ ESB and service provider

The difference from the Service Creation scenario in Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61 is that the message protection policies are no longer point-to-point policies between service consumers and providers, but rather, they are policies that specify the interaction between these components and the ESB.

For example, if the ESB is expecting signed and encrypted messages from a service provider, then the service provider must be aware of this so that outbound messages are correctly prepared before sending. If the ESB then requires communication with one or more service providers to provide a response to the service consumer, it will need to be aware of message protection policies for interacting with those service providers.

Provider policies

Provider policies are policies internal to a particular service provider. Authorization and audit policies are commonly employed to describe the entities able to access the service, and what audit trail is to be generated for future compliance activities. Another example of a provider policy may be business rules within the service, such as *The maximum daily trading limit is \$100,000.*

Service providers may also rely on policies configured in the security services that they consume. For example, an Authentication Service has a local policy for describing account and password validity. Service providers that use that authentication service do not implement these policies (it is likely that they would even be unaware of these policies), but the overall operation of the system relies on these policies being in place to restrict access to the system for entities with expired accounts.

Policy distribution and transformation

High-level, abstract policies described in “Policy administration” on page 108 need to be made available to components requiring the information specified in those policies. The policies need to be translated into a form that those components can understand, since each component will have its own internal representation of the aspects of policy that are important to it. Figure 4-16 on page 110 shows how policy distribution and transformation apply to this scenario.

For example, a centrally-defined security policy for access to application functions may get transformed and distributed as follows:

- ▶ Access control lists, protected object policies, and authorization rules distributed to Tivoli Access Manager.
- ▶ JACC policy distributed to WebSphere Application Server.
- ▶ Web services security management policy distributed to a DataPower security appliance.
- ▶ Permission set to RACF on z/OS®.

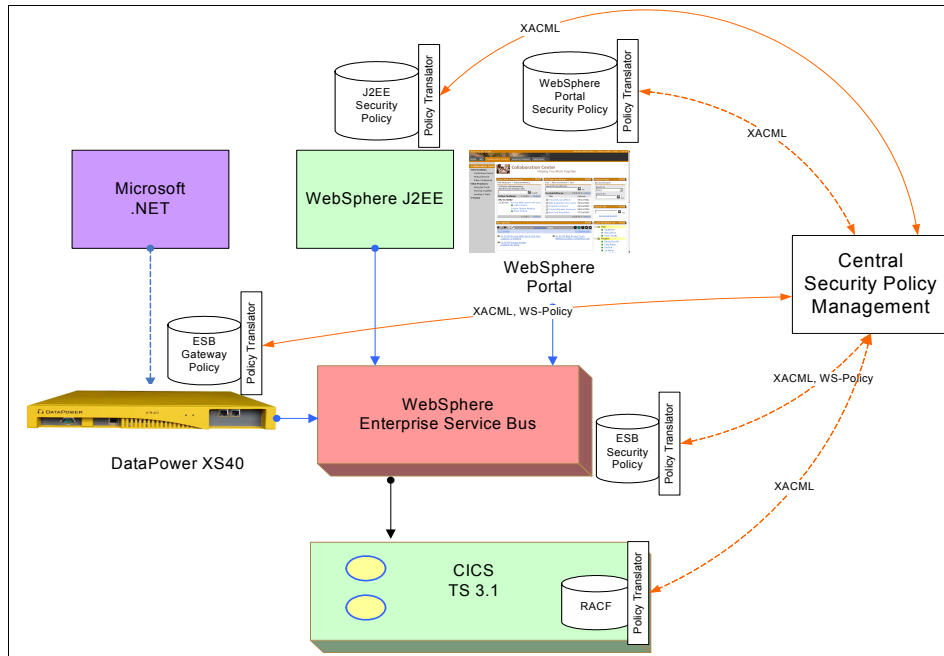


Figure 4-16 Policy distribution and transformation

Policy decision and enforcement

In the Service Connectivity scenario, policy decision points may include:

- ▶ *Common authorization service:* Common authorization services remove the policy decision-making from applications, allowing application developers to focus on business logic. For example, Tivoli Access Manager might be invoked from the ESB and J2EE or Microsoft .NET applications servers to be the Policy Decision Point for all authorization decisions in the service provider implementation.
- ▶ *Common audit service:* Common audit services evaluate their policy for the generation and distribution of audit data. For example, a central data warehouse may be used in the enterprise to store audit records collected across solution components.
- ▶ *Service provider:* While it is preferable to externalize policy decisions from applications, there may be cases where at least some of the policy decision-making logic must reside in the service provider itself. Use of this approach should be minimized wherever possible.
- ▶ *Trust service:* The trust service may perform validation of some security tokens as one of its functions. For example, the trust service may validate the format and integrity of SAML tokens.

Note: For aspects of security policy where a common service is used, the Policy Decision Point is logically consolidated, even if it is physically distributed for reasons of performance and availability.

Policy Enforcement Points are usually distributed across multiple components in the flow of a transaction through the SOA environment:

- ▶ *Service consumer:* In the case of a user accessing the service through a portal, the portal (or a Security Enforcement Point in front of the portal) can act as an enforcement point to determine access to the service before it is even invoked.
- ▶ *ESB:* An ESB may enforce policies related to the protection of messages, based on the results of cryptographic operations for signature validation or decryption.
- ▶ *Trust service:* The trust service may also externalize some of its policy decision making and only act as the enforcement point. For example, authorization decisions may be externalized to a common authorization service, such as Tivoli Access Manager.
- ▶ *Service provider:* Application servers, EIS systems, and so on, enforcement may be provided at the container level or in the application code itself, depending on how policy decisions are invoked (programmatic versus declarative).

Service consumers can also perform some policy enforcement functions, but it is not recommended that this be a substitute for policy enforcement in the ESB or the service provider.

Monitoring and reporting

Based on the information provided within the audit infrastructure and the policy distributed to the decision points, reports can be generated to contribute to the compliance process. Activity reports are also made available for auditing.

4.2.3 Business Security Services

The Business Security Services portion of the IBM SOA Security Reference Model represents a diverse collection of services and capabilities required to provide a verifiably-secure deployment environment for a SOA solution. Most of what is discussed in this section complements a SOA deployment in the same way that it complements other application deployment approaches.

Figure 4-17 shows the list of Business Security Services from the SOA Security Reference Model. We will discuss each of these security services in turn, and show how they can be used within the SOA Foundation Service Connectivity scenario.

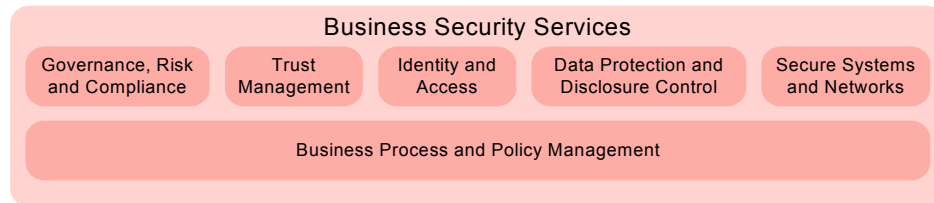


Figure 4-17 Business Security Services from the SOA Security Reference Model

Governance, risk, and compliance

A SOA Governance Board would likely:

- ▶ Interact with the SOA Governance Board of external partners connecting to the services being provided to establish an effective working relationship between the two organizations.
- ▶ Specify policies for message protection of messages that cross the enterprise boundary.

As a result of the risk management processes, compliance activities might include:

- ▶ Having personnel managers validate the continuing business need for their employees to have access to the set of services that have been provided to them.
- ▶ Generating reports to demonstrate compliance with a regulation such as the Health Insurance Portability and Accountability Act (HIPAA), the Sarbanes-Oxley Act (SOX), or the International Convergence of Capital Measurement and Capital Standards (Basel II).

Trust Management

From a business viewpoint, Trust Management includes the liability and legal aspects around the services.

At the technology level, Trust Management may define:

- ▶ Standards for the strength of cryptographic ciphers to be used when accessing the ESB
- ▶ Standards for security tokens associated metadata
- ▶ Key management for X.509 certificates for the ESB and service components

Identity and Access

This section describes how to apply the Identity and Access services defined in 4.2.1, “IT Security Services” on page 95 to manage the identities in the environment, their attributes, and access privileges.

Identity feeds from HR systems are used to populate the user repositories in the solution, such as the common authentication service, application, and EIS user repositories. Attributes of the users, including their roles or group memberships, are based on identity provisioning rules.

Business requirements may also dictate manager intervention to approve user accounts or to periodically re-validate a user’s accounts to meet regulatory compliance objectives.

To distribute the management and administration cost of some functions, user self-care via the WebSphere Portal could be implemented. Functions such as password synchronization and update of personal information are typical user self-care functions. Enrollment and initiation of request for business services is an increasingly popular approach, supported by the manager workflow described in the previous paragraph.

Data protection and disclosure control

Data protection management uses the services described in “Confidentiality and Integrity Services” on page 104. Data protection management identifies the resources that need protection and the controls required on those resources. For example:

- ▶ Where possible, cryptographic key stores should be located on dedicated hardware for secure storage.
- ▶ Where cryptographic key stores on the file system have to be used, they should be protected with operating file system permissions, and stored on an encrypted file system.
- ▶ An operating system hardening solution should be used to protect service provider machines, and best practices be employed for identifying the files to be protected, based on the type of application server used.

Disclosure control uses the Authorization and Privacy services introduced in 4.2.1, “IT Security Services” on page 95. An organization may publish its privacy policy for users to review prior to subscribing to services that it provides, allowing them to make an informed decision before opting-in to using these services. A reporting mechanism should also be available to provide data on disclosure control to the compliance function described in “Governance, risk, and compliance” on page 112.

User consent is almost always a component of a disclosure control implementation. For example, a user's consent may be required before a user's accounts are federated. Another aspect of disclosure control might be that a user has access to a protection that shows what personal information the organization is retaining about them.

Secure systems and networks

Firewalls are one of the most fundamental mechanisms for isolating networks in the deployment environment. As shown in Figure 4-18, firewalls could be used to separate:

- ▶ The outside zone from the demilitarized zone (DMZ) where the point of contact resides
- ▶ The DMZ from the production zone where the ESB and the service provider reside
- ▶ The production zone from the internal network where the enterprise information system
- ▶ The DMZ, production and internal zones from the management zone where the security services reside

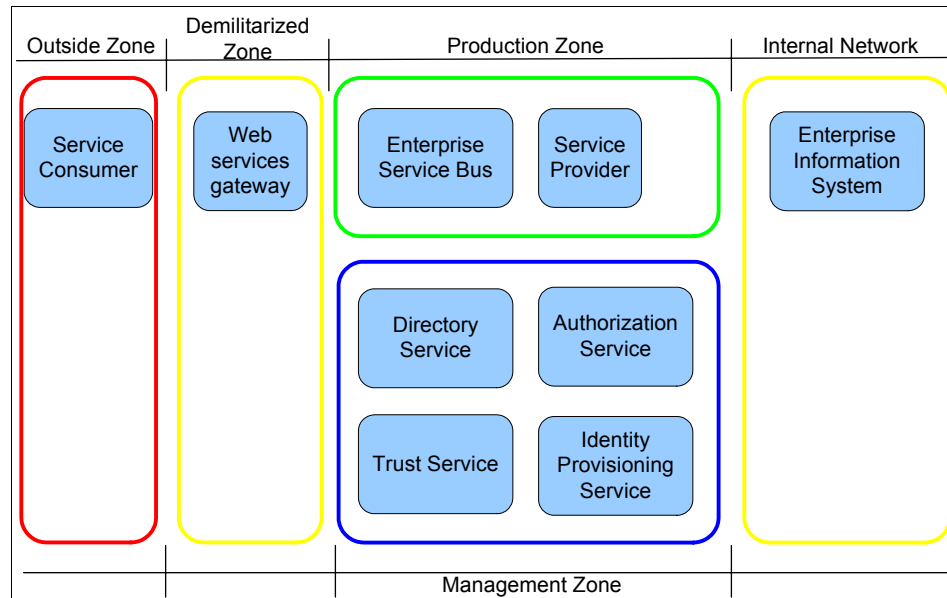


Figure 4-18 Typical logical network diagram for the SOA Foundation Service Connectivity scenario

Note: For additional detail on secure network topologies, consult Chapter 2, “Common security architecture and network models” and Appendix A, “Method for Architecting Secure Solutions”, in the IBM Redbook *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014.

Host and network-based intrusion detection systems (IDS) can be used to aggregate logs from a variety of devices to correlate and identify potential security issues. The IDS should be integrated with a monitoring and alerting system so that an escalated response to an incident can be initiated when appropriate.

Security devices can also provide protection against XML viruses by providing strong schema validation and examination of Web service messages for well-known exploits.

Managing the security of the deployment environment where business solutions can be deployed and hosted is important. There are a set of tools that help protect infrastructure servers, systems, and networking resources from security threats. Solutions in this category provide protection against viruses, hackers, and misuse by internal users.

As availability relates to security, the performance and availability indicators that need to be monitored have to be defined. For example, the unsuccessful login attempts and some response times of the system can be monitored.

Note: More information about monitoring activities within SOA are available in *IBM Tivoli Composite Application Manager V6.0 Family: Installation, Configuration and Basic Usage*, SG24-7151.

Business Process and Policy Management

In the Service Connectivity scenario, some aspects of Business Process And Policy Management include:

- ▶ Managing compliance activities related to authorizing access to services based on periodic revalidation of the business need.
- ▶ Establishing policies for access to services based on the incoming user’s organization and role.
- ▶ Process around managing trust relationships with external service consumers.
- ▶ Classifying choreographed applications and data so that appropriate data protection and disclosure control can be implemented.

4.3 Summary

The application of the IBM SOA Security Reference Model to the SOA Foundation Service Connectivity scenario has been described in this chapter. There are many different security issues to contend with, and these must be applied across the different components of the scenario. This chapter is therefore usable as a checklist in determining where security can be applied to a real customer environment.



IBM SOA Foundation Service Aggregation scenario

This chapter discusses the application of the IBM SOA Security Reference Model, defined in Chapter 2, “Architecture and technology foundation” on page 17 to the IBM SOA Foundation Service Aggregation scenario.

The chapter provides an overview of the Service Aggregation scenario and highlights the components involved. Then it provides guidance to apply the SOA Security Reference Model to the scenario, covering the following areas:

- ▶ IT Security Services
- ▶ Security Policy Infrastructure
- ▶ Business Security Services

5.1 Scenario overview

In this section, an introduction to the scenario, the actors, and the components involved is provided. The scenario is then detailed from different perspectives to make the understanding of the security requirements identified easier.

5.1.1 Overview of the Service Aggregation scenario

The goal of this scenario is to provide access to a set of services for a user through a common interface.

The business value for this scenario is to reuse existing information and data through a common entry point to improve user productivity.

Services can be internal (existing applications or exposed functions) as well as external, meaning they are provided by a partner. These services can be provided on both sides through a portal, for example.

The services are aggregated through a corporate portal in this scenario, using a role based model to provide personalized content to the users.

The user may also access an external portal that aggregates some services provided by a partner. Figure 5-1 provides an overview for this scenario.

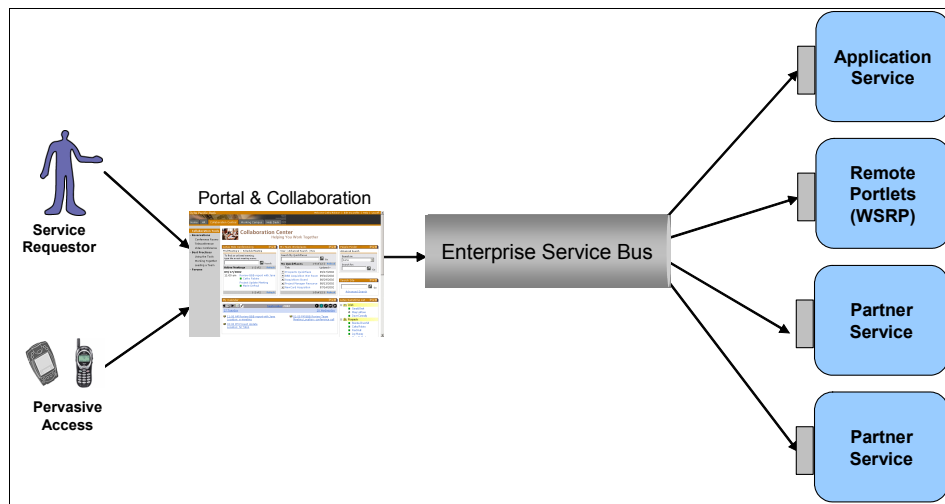


Figure 5-1 Service Aggregation scenario overview

The following actors and components are identified for this scenario:

- ▶ *User*: The user can be an internal employee as well as an external user accessing a set of services through the company portal. This user can access the company portal as well as external portals using different channels: a Web browser, a rich client interface, or a mobile device.
- ▶ *Company portal*: The company portal provides aggregation and collaboration services to users through a common interface. This includes access to internal services as well as external services provided by other companies. The company portal may also aggregate some content of external partners also provided through a corporate portal.
- ▶ *Internal service provider*: A service provided by internal components.
- ▶ *External portal*: A portal provided by a third party. This portal can be accessed by the user through a Web browser or any other device. Different services are aggregated in this portal.
- ▶ *External service provider*: A service provided by the third party. The service can be accessed through the external company portal and directly from the company portal itself, as shown in the Figure 5-1 on page 118.

This scenario can be considered from several perspectives to detail the security requirements that apply:

- ▶ The Web single sign-on perspective focuses on the user based interactions with the different portals involved in this scenario.
- ▶ The Web services perspective concentrates on the integration of the different services provided and the interactions between these services.
- ▶ The provisioning perspective brings out the flows and the required synchronization events for the different accounts used by the user in the scenario.

The Web single sign-on and the Web services perspectives are detailed below. The provisioning perspective is highlighted in “Identity Services” on page 124.

5.1.2 Web single sign-on perspective

This section focuses on the single sign-on capabilities provided to the user in the scenario.

Using a Web browser, the user accesses internal applications. Some of these applications are accessed through the company portal while others are not. Single sign-on has to be provided to the user for all these applications.

The user also accesses some applications provided by a third party and needs to have single sign-on to these applications as well. In this case, *federated single sign-on* is used.

Figure 5-2 details the interactions for the Web single sign-on perspective of the scenario.

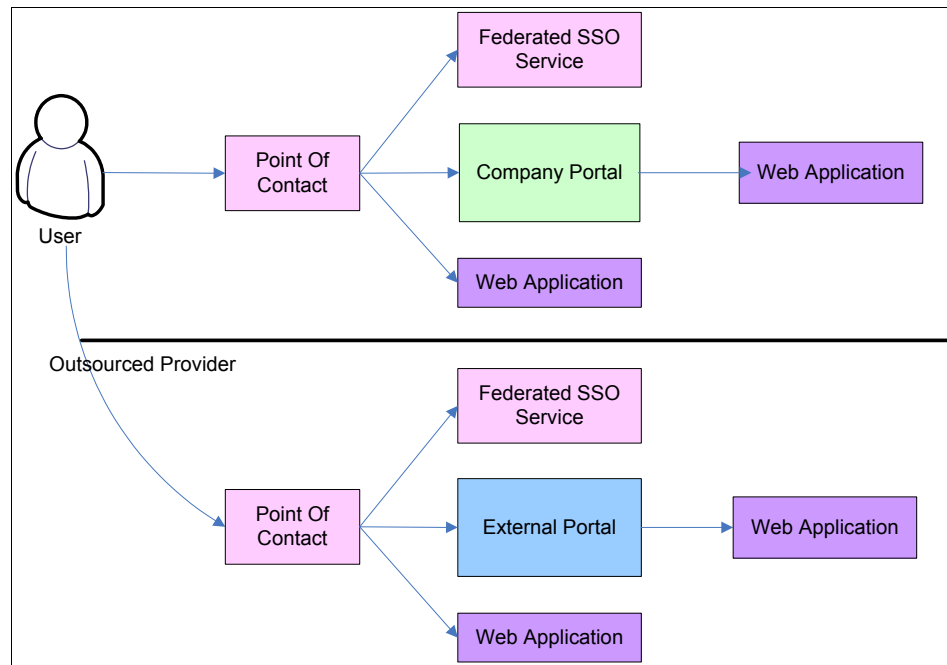


Figure 5-2 Web single sign-on perspective for the Service Aggregation scenario

For this perspective, the following components are introduced:

- ▶ *Point Of Contact*: This service provides an entry point to the system for user Web-based interactions and provides single sign-on capabilities to the internal applications. This includes the company portal as well as any other internal Web applications that are not aggregated through the portal.

From a technical point of view, a reverse proxy or a plug-in for Web servers is used as the point of contact.

- ▶ *Federated SSO Service*: This service provides the core capabilities to support federated single sign-on for users between a company and its partners.

This includes the support for the different federation protocols and the ability to generate, validate, or exchange the appropriate security tokens used between the federation partners.

These services are also required for the partner infrastructure. The remainder of this section focuses on the company infrastructure as the federation protocols ensure interoperability.

Note: More information about these services and the architecture of Federated Identity Management solutions can be found in the IBM Redbooks *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014 and *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394.

- *Web application:* The Web applications accessed by the user. This includes applications aggregated through the company portal as well as existing ones that are not accessed through the common entry point.

Note: In order to simplify the understanding and to stay focused on the core requirements of the scenario, the point of contact will be assumed to be embedded in the company portal, unless indicated.

5.1.3 Web services perspective

Considering the Web services perspective, the user accesses the company portal that aggregates content from various sources. This includes internal services as well as external services. Figure 5-3 depicts this perspective.

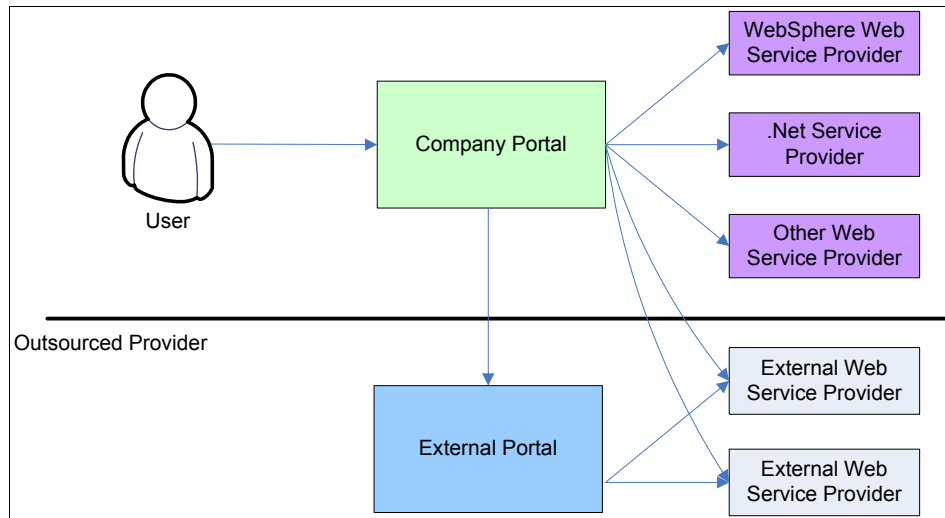


Figure 5-3 Web Services perspective for the Service Aggregation scenario

The services accessed can run on different platforms. For example, a WebSphere Web service is provided and accessed through the portal as well as a Microsoft .NET one. A service already exposed can be reused in this scenario, such as the one exposed in Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61.

External Web services may also run on different platforms. They are accessed through the company portal.

Finally, the company portal may access services or functions already aggregated into the external provider portal. This can be achieved for example using the standard *Web Services for Remote Portlets* (WSRP). This standard defines a way to provide services for companies running portals.

Note: The complete specification for the Web Services for Remote Portlets is available at <http://www.oasis-open.org>.

Another approach for this perspective can be to consider the external users (for example, employees from the outsourced provider) accessing the company services exposed through their corporate portal. Such a case can be considered as an application of the Service Creation scenario defined in Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61.

5.2 Applying the IBM SOA Security Reference Model

Figure 5-4 on page 123 restates the IBM SOA Security Reference Model with the three main components: IT Security Services, Security Policy Infrastructure, and Business Security Services.

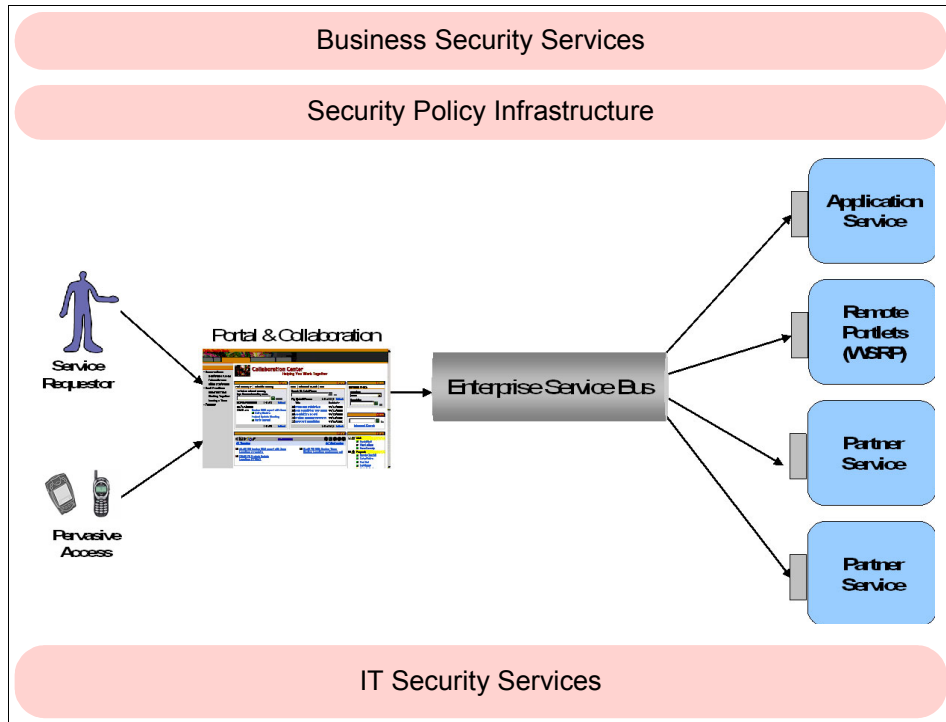


Figure 5-4 Applying the SOA Security Reference Model

This section provides guidance on how this reference model can be applied to the scenario. It provides some examples of common use cases for the scenario, but is not exhaustive. Different business requirements than the ones detailed below can lead to different architectures.

5.2.1 IT Security Services

In this scenario, the IT Security Services provide a set of common services for the different components involved in the architecture.

Figure 5-5 provides a list of the IT Security Services. Each of these security services is then discussed in the scope of this scenario.

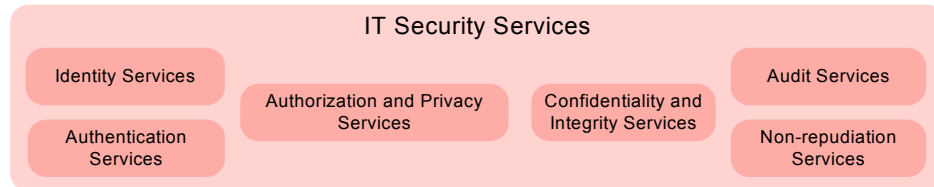


Figure 5-5 IT Security Services from the SOA Security Reference Model

Identity Services

These services provide the core infrastructure for managing user accounts between the different systems aggregated in the scenario. Identity Services include identity foundation, identity provisioning, and identity federation.

Identity foundation

Different user repositories are identified in this scenario. Figure 5-6 on page 125 provides an overview of these different repositories and the account identifiers used for the same company employee, Joey Smith, both for internal accounts as well as external accounts.

For each account, an identity is defined and identity information is attached to the person. Synchronization between these accounts is a key requirement, as the person needs to have the appropriate permissions defined in the identity repositories to access the different services.

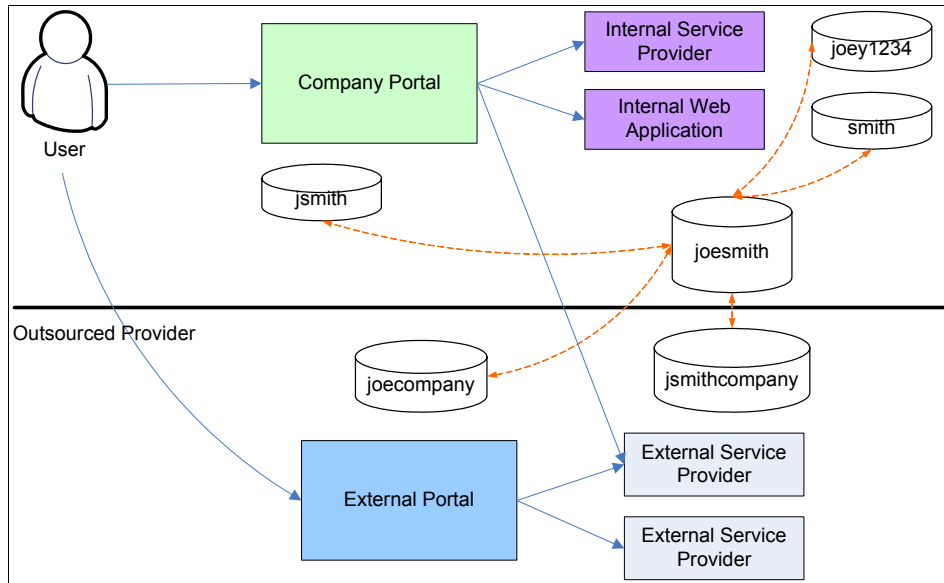


Figure 5-6 Identity Foundation for the Aggregating User Interface scenario

Internal repositories are identified for the company portal (an LDAP directory or a database can be used) and then for any service provided (directories, databases, or any custom repository may be used). Finally, an enterprise repository can be used as the company repository.

When considering the external provider, the company employee has an account for accessing the portal and may have additional accounts defined in the appropriate service repositories.

Synchronization has to be considered between all these repositories. For example, if a user changes his phone number or updates any information he is allowed to change, this change has to be propagated to the other repositories using this information.

Identity provisioning

Using a provisioning solution in this scenario can help in managing the identities both inside and beyond the company boundaries. This solution is responsible for creating, managing, and deleting the accounts in the appropriate repositories.

Such a solution reduces administration complexity related to the identity management and enforces security as it leverages the company security policies (as defined in “Policy administration” on page 139). It can rely on a role-based model, meaning the identities are provisioned based on roles defined for the enterprise.

For example, a banker has an account created for a mainframe application and is added to a specific group of users within the mainframe identity repository (for example RACF) so that he has access to the mainframe application with the appropriate permissions.

Moreover, if the employee changes from one role to another or leaves the company, the provisioning solution ensures that his accounts are updated or removed from the systems automatically.

Figure 5-7 introduces the identity provisioning solution to the scenario.

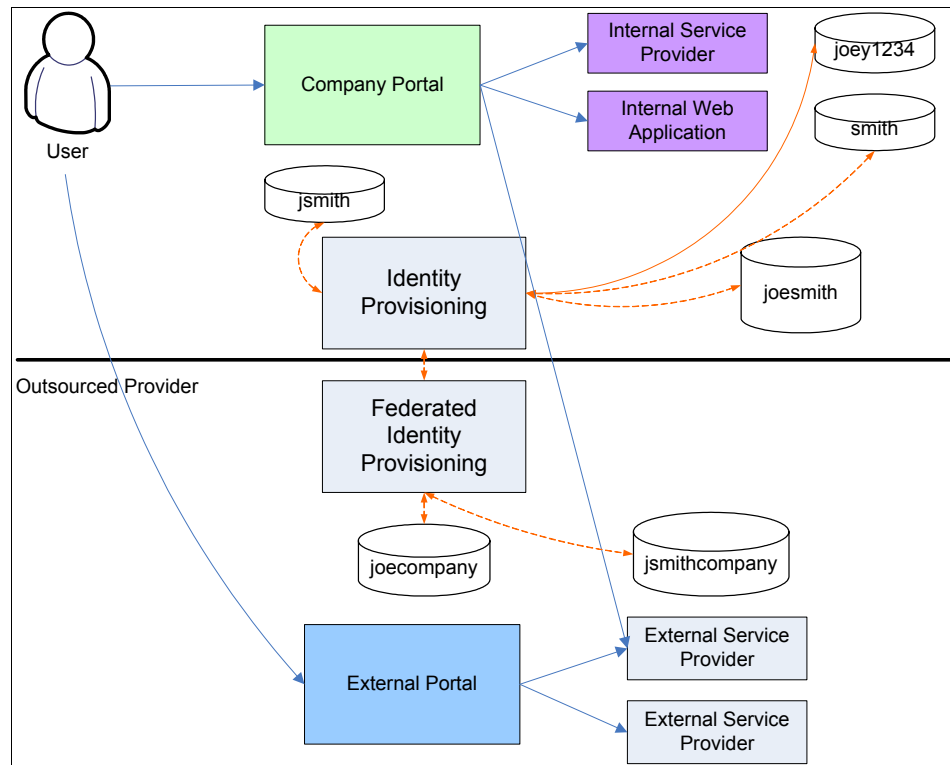


Figure 5-7 Identity Provisioning for the Service Aggregation scenario

As shown in Figure 5-7, the identity provisioning solution can be extended to cross the company boundaries. Federated identity provisioning allows the provisioning of user accounts outside the company.

The accounts can be provisioned for federated single sign-on between the company and the outsourced provider for the users. In this example of federation relationship, the company acts as an identity provider and the outsourced

provider as a service provider. The use case is detailed in “Identity federation” on page 127.

Identity federation

Several use cases can be defined to illustrate the need for identity federation services in this scenario.

The first use case is the single sign-on between the point of contact and the internal Web applications, such as the company portal. This use case applies if these components are separated, as detailed in 5.1.2, “Web single sign-on perspective” on page 119.

In this case, the point of contact and the company portal (or any other internal Web application) have their own identity token format and there is a need to securely transfer the identity from the SSO point of contact to the company portal.

Note: Such a use case has already been designed and detailed in some IBM Redbooks, such as *Develop and Deploy a Secure Portal Solution*, SG24-6325 or *Enterprise Business Portals II with IBM Tivoli Access Manager*, SG24-6885. The integration for this scenario can rely on the architecture and techniques described in these references and is not further described in this section.

Another common use case is the access to an internal service provided within the company. This case has been detailed in Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61 where existing functions are exposed, either directly or through service components.

The trust service component has been introduced to validate, exchange, or generate security tokens. The architecture for this case in this scenario is shown in Figure 5-8.

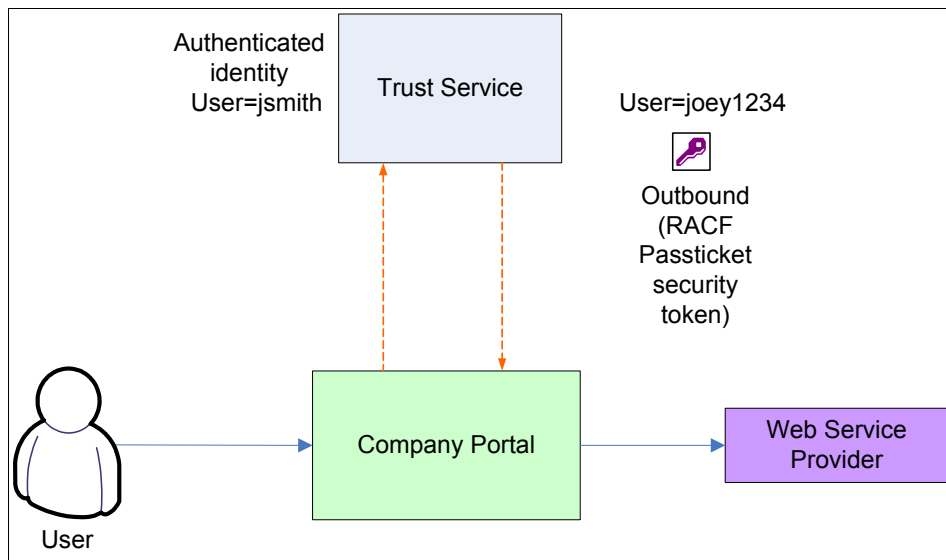


Figure 5-8 Identity federation for the Service Aggregation scenario: Access to an internal Web service

In this scenario, the user authenticates to the company portal using a user identifier and password for example (jsmith). In order to access the exposed Web service (a CICS application, for example), the portal needs to map the security token into a valid representation for the receiving system (a RACF Passticket with the user identifier joey1234, for example). This exchange is done using the trust service.

Another use case is the access to an external service provided by a partner through the company portal. Figure 5-9 on page 129 shows the use case.

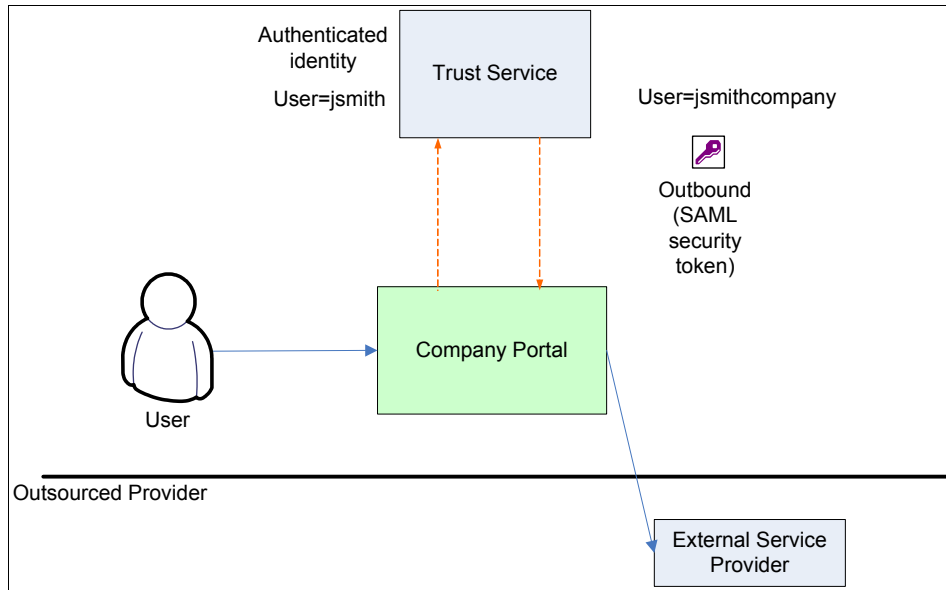


Figure 5-9 Identity Federation for the Service Aggregation scenario: Access to an external Web Service

In this case, the trust service is invoked to map a security token for a valid representation for the external service provider. In this scenario, this token can then be a user name token or a SAML assertion generated for the user. The external service provider is then able to validate the SAML assertion to map it to the jsmithcompany user account.

Finally, another use case introduces the ability to provide federated single sign-on to the third party portal. Figure 5-10 details this scenario.

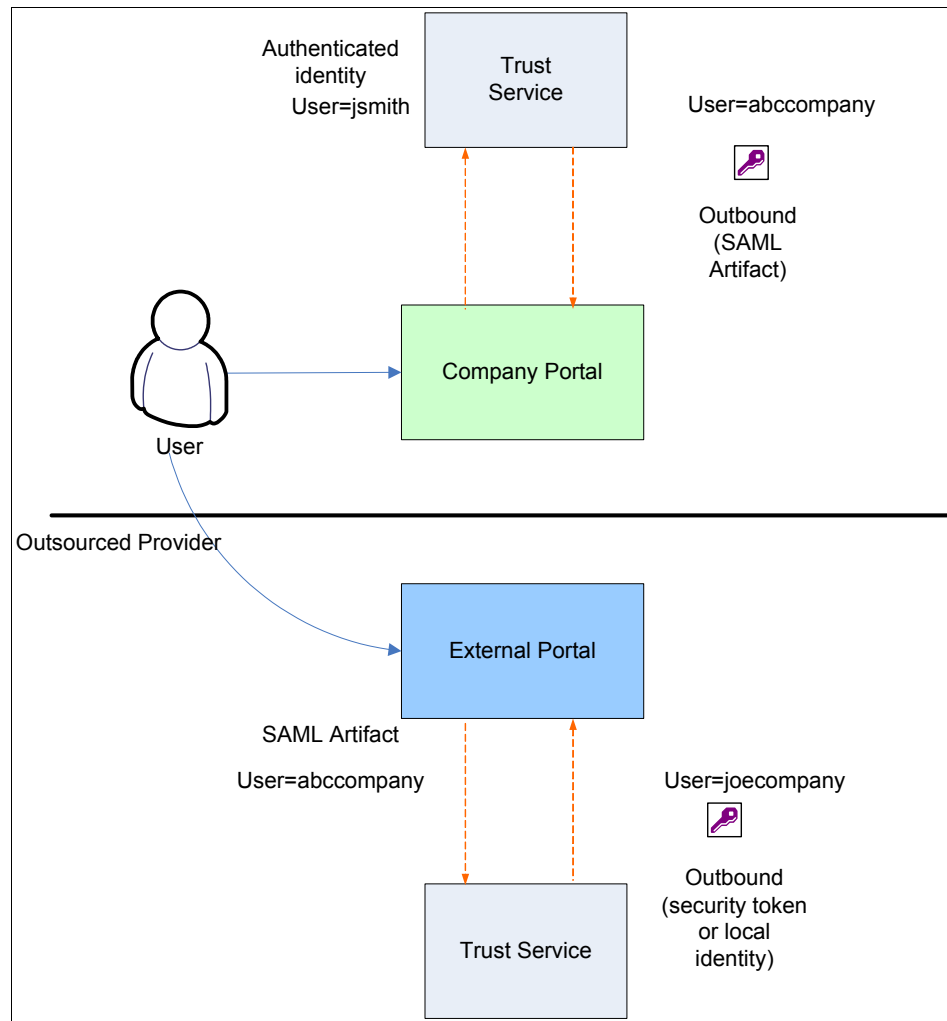


Figure 5-10 Identity federation for Service Aggregation scenario: Access to an external service with federated single sign-on

In this scenario, the user accesses his company portal. The trust service generates a security token for the partner based on the user identity and the information required for the partner. This token generation is generally provided by a federated single sign-on service (as identified in 5.1.2, “Web single sign-on perspective” on page 119).

The partner receives this token and can invoke its trust service to validate the token and map it to a local identity.

Authentication Services

In this scenario, Authentication Services have to deal with different domains, different platforms, and different services. As a consequence, there are several enforcement points where authentication services are provided.

The first authentication challenge takes place when the user accesses his company portal. This authentication is generally achieved with a user identifier and password, but strong authentication methods can be combined to provide additional security. This user identifier and password is checked against the repository used by the point of contact. Figure 5-11 depicts this use case.

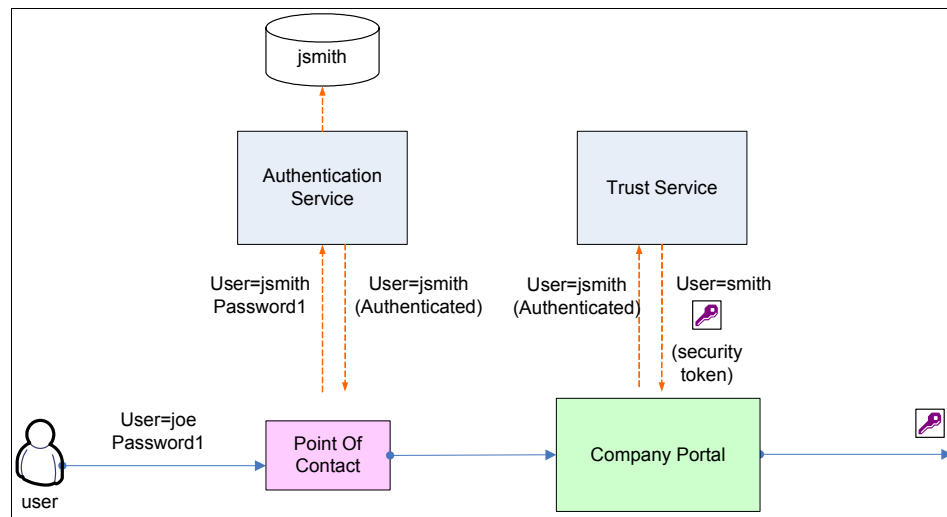


Figure 5-11 Authentication Services for Service Aggregation scenario: company portal authentication

This Authentication Service can be provided with a point of contact in front of the company portal as defined in 5.1.2, “Web single sign-on perspective” on page 119 and shown in the Figure 5-11.

The use of a dedicated component in front of the company portal is useful to externalize the Authentication Service from the portal itself and to provide a dedicated Policy Enforcement Point for the authentication policies.

The point of contact can provide these services to other Web applications as well. It enforces security providing a first level of defense within the infrastructure.

It can enforce the password policies for the platform. For example, a user account may be locked after three unsuccessful login attempts within a short period of time.

Session management services are also provided. Finally, it can provide flexible authentication as well as support for strong authentication methods and re-authentication based on the company policies.

In case the point of contact is used in the infrastructure, a trust relationship is established with the company portal. The SSO point of contact authenticates the user and provides identity information to the portal. It is common that both components use the same user repository.

Once the user is authenticated, the point of contact grants access to the portal. The portal now needs to pass appropriate identity information to back-end applications. In this example, the portal aggregates content from various sources, internal Web applications, and internal Web services, as well as external content.

The trust service is used to retrieve the relevant information for the targeted service. This includes the security token information (the correct user identifier or certificate) and the security token format, which may change depending on the service accessed. In this example, a security token for the user identifier smith is retrieved.

The portal then passes this information to the targeted service. Depending on the channel used, this information can be provided in different manners, as a Web services security token or using HTTP headers for a Web application.

When receiving the request, the service provider (either internal or external) needs to validate the token. It can invoke the trust service that uses an Authentication Service to check the credentials provided. This can be a user name / password check against a registry, a SAML assertion, or a digital certificate validation.

Figure 5-12 on page 133 provides an example of the authentication performed at the service side.

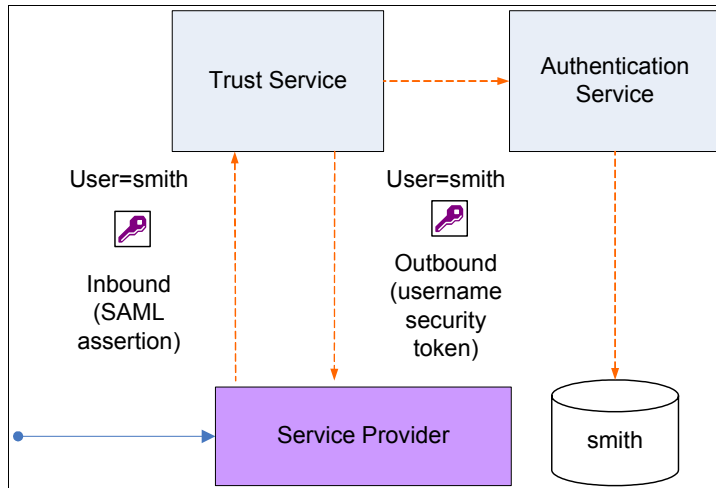


Figure 5-12 Service provider authentication

It also applies to the federated single sign-on use case. The local company token is exchanged for a federated SSO token that is validated by the service provider trust service. In some cases, the token provided by the company can be an artifact. A back channel secured communication between the service provider trust service and the company trust service is then used to retrieve the complete security token from this artifact.

Note: More information about the federated single sign-on protocols and specifications are provided in Appendix C, “Security standards and technology” on page 371.

Authorization and Privacy Services

Authorization is enforced at different levels in the scenario as shown in Figure 5-13.

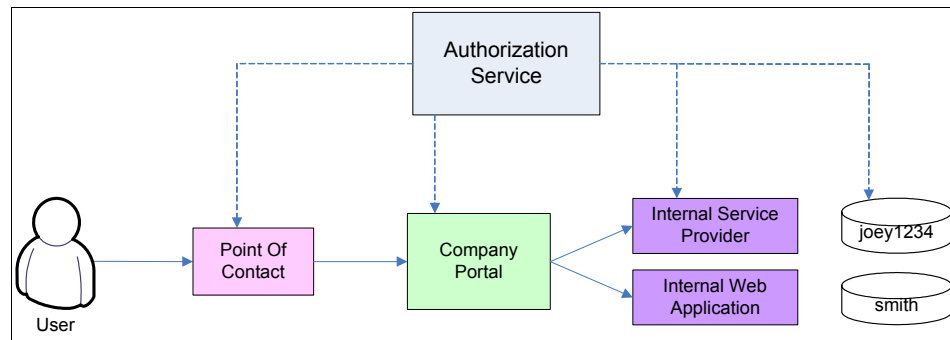


Figure 5-13 Authorization Services for Service Aggregation scenario

The following components provide the authorization enforcement in this scenario:

- ▶ *Point of contact:* The point of contact provides a first level of defense for the company portal and the other internal Web applications.

Coarse-grained authorization can be provided here to prevent access from an untrusted network or to restrict access to the business hours, for example. Finer-grained authorization rules can also be defined to allow groups of users to access the portal or some of its features.

- ▶ *Company portal:* The company portal can provide finer-grained authorization services for the users. This can include role-based permission to allow the users to access personalized content.

For example, a user may be allowed general access to internal services based on his profile although he is not authorized to perform certain operations on these services. Declarative security should be used as much as possible to separate the authorization policies from the application itself.

- ▶ *Applications:* The various service providers involved in the scenario, either internal or external, may have their own mechanisms and authorization repositories to render authorization decisions.

An existing system, such as CICS, may use RACF to store authorization rules, while another service may use a proprietary repository or a database to store these rules. In any case, the services can provide finer-grained authorization controls based on the application policies.

- *Data*: Rules to authorize the access to the information can be considered there. This can include access control lists on an LDAP server or the permissions within a database.

Such authorization rules are generally provided by the component itself, and they are difficult to externalize.

As part of the authorization decision, privacy may be considered in this scenario to provide access to the resources in a way that is consistent with company policies.

Confidentiality and Integrity Services

Data located in the different repositories needs to be protected. Access to the machines hosting these repositories needs to be secured and the passwords have to be encrypted.

Information in transit also requires additional security. Different messages are involved in this scenario and they can be secured in different ways, depending on their nature and the endpoints involved.

Figure 5-14 provides an overview of the messages requiring transport level security (done with SSL) and the ones requiring additional message level security (achieved using WS-Security) in this scenario.

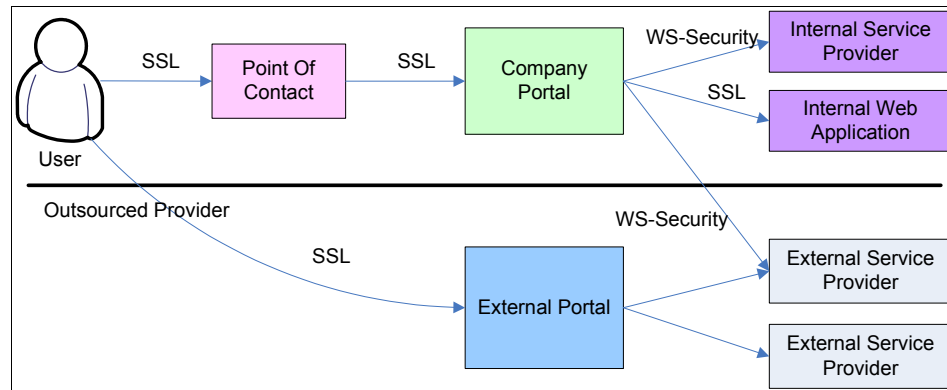


Figure 5-14 Message protection services for Service Aggregation scenario

The use of message protection services is specific to each organization. It depends on the network trust and the data classification inside an organization's network and so may be different from one organization to another.

In this scenario, the message protection services may apply as follows:

- ▶ Transport level security is generally achieved using SSL and can be used for the following flows:
 - Any HTTP based interactions, from the user to the point of contact or from the user to the external portal.
 - Internal communications, for example, the request coming from the point of contact to the company portal or internal access to Web applications.
- ▶ Message level security can be used in addition to transport level security to provide additional security for the following messages:
 - Access to an internal service from the company portal. Depending on the EIS requirements, the appropriate security token can be provided (a user name token using a RACF Passticket for a CICS transaction for example) and the message may contain additional encryption and signatures.
 - Access to an external service from the company portal. A common use case for this scenario is to use a SAML assertion to access the external service provider. This assertion can be signed and encrypted.

Audit Services

Audit events are generated in this scenario to keep track of any relevant security activity. This tracking allows the generation of reports to check the compliance of the system regarding the security policies defined for the company.

When having a look at the services provided in this scenario, it is important to provide Audit Services for the following activities:

- ▶ *Identity Services*: Identity provisioning events, such as provisioning of new accounts, including the related workflow activities, are audited in this scenario.

For example, the company needs to be able to know which accounts have been created on a dedicated system and if the process of getting the manager approval to create these accounts has been followed. Auditing the appropriate provisioning events thus allows the generation of a report (in a chosen format) containing all the accounts created with their approvers for a period of time.

The same kind of reports can be useful for federated provisioning. Generating reports dedicated on a per partner level can then be used for compliance reporting on the business relationship between the company and its partners.

Identity federation events are also audited. The token exchanges are audited and the original or local identity is tracked.

- ▶ *Authentication Services*: Authentication generates audit events. These events can be stored locally to the component performing the authentication or aggregated to a common audit infrastructure.
- ▶ *Authorization and Privacy Services*: Authorization decisions can also generate audit events. These events can be provided for several purposes depending on the component performing the authorization and the type of authorization.

Coarse-grained authorization events can be used to monitor the network security (for example, IP address control at the point of contact level) while finer-grained authorization performed at the application level is used for auditing the application itself.

- ▶ *Confidentiality and Integrity Services*: Additional audit events can be generated for message protection services, including signature validation events that can be used in order to ensure integrity.

The audit infrastructure also relies on the other core Security Services defined in the SOA Reference Model. For example, the audit data needs to be protected as with any other repository within the system.

Non-repudiation Services

The Non-repudiation Services are in place so that there is evidence that a transaction has taken place. There are two aspects, as previously described:

- ▶ Protect the recipient from the false denial by an originator that the data has been sent.
- ▶ Protect the originator against the false denial of a recipient that the data has been received.

From the point of view of the service provider in the Service Aggregation scenario, the portal first receives the user request, and accesses other service providers to fulfill those requests. In this case, the portal may invoke the Non-repudiation Services, using evidence such as the user's authentication to the portal, and that they are accessing the portal over a secure channel, such as HTTPS. The service providers accessed by the portal may use the Non-repudiation Services to record the transaction coming from the portal. A combination of multiple non-repudiation records may be required to associate a user's request with the events that occurred on a set of service providers.

From the point of view of the service consumer in the Service Aggregation scenario, the important evidence to record is that the portal has responded to the request. Unlike the Service Creation scenario, access to the service providers is not direct, so evidence for the non-repudiation record is likely to be based upon the secure channel between user and portal. A non-repudiation record can then be created at the service consumer by calling on the audit service, to record the

signed data response with a time stamp to indicate when the response was received.

Summary

Table 5-1 summarizes the requirements that can be applied to the Security Services in this scenario.

Table 5-1 A checklist of security services from the IBM SOA Security Reference that can be applied to the Service Aggregation scenario

Security Services	Application for the Service Aggregation scenario
Identity Services	<ul style="list-style-type: none"> ▶ Identity foundation ▶ Identity provisioning ▶ Identity federation
Authentication Services	<ul style="list-style-type: none"> ▶ User ▶ External user ▶ Company portal ▶ Internal service provider ▶ External service provider
Authorization and Privacy Services	<ul style="list-style-type: none"> ▶ Point of contact ▶ Company portal ▶ Service provider ▶ Service provider data
Confidentiality and Integrity Services	<ul style="list-style-type: none"> ▶ Data in repositories ▶ User to point of contact ▶ Point of contact to company portal ▶ Company portal to internal service ▶ Company portal to external service ▶ User to external company portal
Audit Services	<ul style="list-style-type: none"> ▶ Point of contact ▶ Portal ▶ Service provider
Non-Repudiation Services	<ul style="list-style-type: none"> ▶ Service consumer ▶ Portal ▶ Service provider

5.2.2 Security Policy Infrastructure

Different components make up the Security Policy Infrastructure, as defined in Figure 5-15.

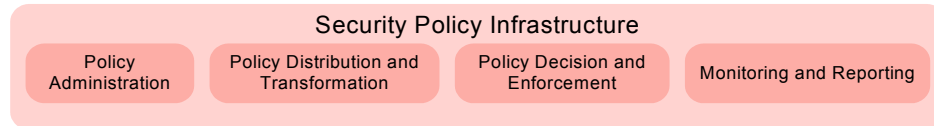


Figure 5-15 Security Policy Infrastructure from the SOA Security Reference Model

As part of the management activities, several policies have to be defined. These policies cover the interactions between a service consumer and a service provider, as well as the provider internal policies and service metadata.

In this scenario, several enforcement points rely on decision points to render security decisions. The different security policies need to be provided to these decision points.

Policy administration

Different types of policies are identified in the SOA Security Reference Model and in this scenario.

Message protection policies

Several service providers interact in this scenario and provide their policies to the service consumer (the company portal, in this example). These policies include the endpoint information to reach the services. For the external service provider, intermediaries may be used.

Message protection can be used in this scenario to access the external services. Internal services exposed to partners may also use this type of security. Digital signature and encryption information (supported algorithms) need to be provided as well as the security token to be used.

These policies may be provided directly to the service consumers or be made available within a service registry.

In this scenario, a SAML token can be used to access the external service provider from the portal.

Provider policies

The different services aggregated in this scenario run on different platforms and use different environments so they have their own internal policies.

For example, an internal service providing access to a CICS transaction relies on a RACF system to authenticate and to authorize the user requests. The access management policies related to the service thus need to be defined in this repository.

A WebSphere internal service relies on declarative and programmatic security to make the same type of decisions, using a different LDAP directory as the external user repository.

Finally, a Microsoft .NET service provider may rely on an external authorization manager queried using XACML to render access control decisions.

No assumption is made on the external service provider internal policies. The company portal acts as a service consumer and thus may not know them, relying on the interaction policies for this provider to request it.

Policy distribution and transformation

The different policies defined in this scenario need to be transformed into a generic format and provided to the different Policy Decision Points defined in the architecture.

Figure 5-16 on page 141 shows the distribution of these policies from the central security policy management infrastructure in this scenario, both for the company and for its partners.

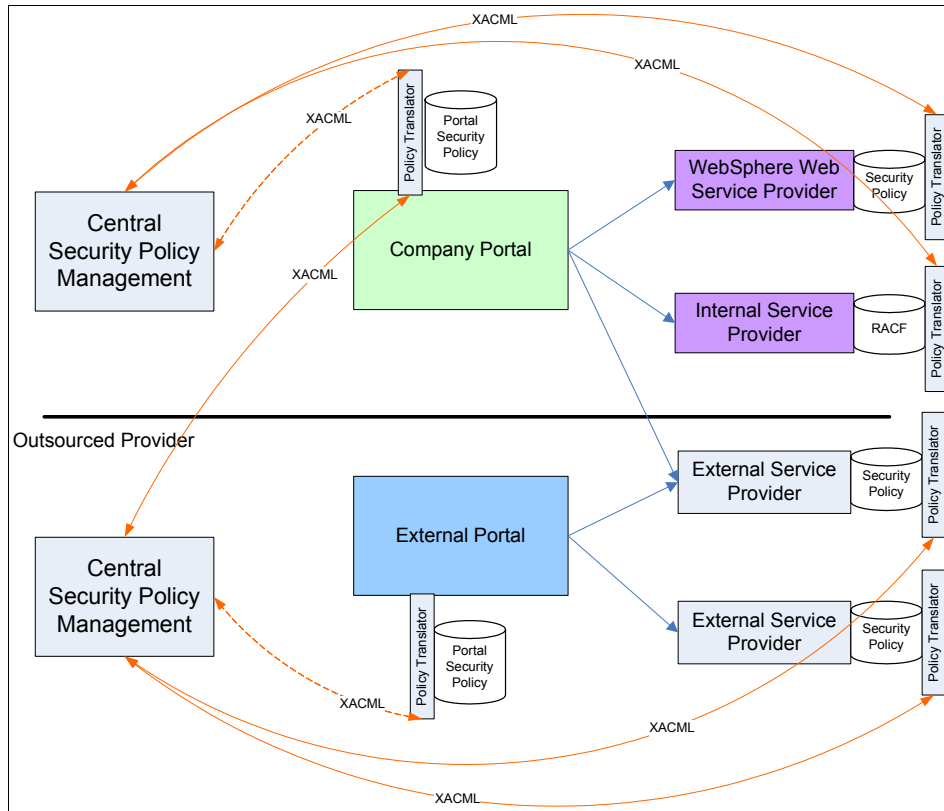


Figure 5-16 Policy Transformation and Distribution for the Service Aggregation scenario

The policy transformation and distribution can be considered for two types of components:

- ▶ *Internal components:* This includes the different service providers and the company portal as a service consumer in this scenario.
- ▶ *External components:* The interaction policies have to be provided to the different service consumers, including the external ones.

Policy decision and enforcement

The enforcement and decision points used within this scenario can be the following:

- ▶ *The point of contact:* The first level of defense can be provided by a point of contact as defined in “Web single sign-on perspective” on page 119. This Policy Enforcement Point can rely on a separated Policy Decision Point to provide some security services or use common services (such as an Authentication Service, a trust infrastructure, or common auditing services).

The capabilities provided include the following:

- *User authentication*: In this scenario, internal employees authenticate using a user name / password. Using the Authentication Services, this user name / password validity is checked against a registry.
- *Password policy enforcement*: The company may decide to lock the user accounts for a certain amount of time if they submit an incorrect password.
- *Session management*: The enforcement point checks if the user has a valid existing session and manages its life cycle based on the policies. For example, one user may not be allowed to have two open sessions at the same time and inactivity can be set to a fixed period.
- *First level of authorization*: This can include network address control to allow only internal users to access a set of services through the company portal.
- *Auditing services*: The different events related to the access in general or the successful or unsuccessful authentication have to be tracked for further auditing.
- ▶ *The company portal*: The portal can in turn provide an enforcement point for the requests. The decision point used for the portal also relies on common services provided within the infrastructure, such as the trust service. The capabilities include the following:
 - *Identity mapping*: The trust service is used to obtain the appropriate security token from the local identity, depending on the service accessed.
 - *Authorization*: Finer-grained authorization on the portal resources can be achieved.
 - *Auditing services*: The access to the different services have to be tracked.
 - *Add message level security*: When using message level security for access to an external service, the trust infrastructure provides the set of services related to digital signature and encryption.
- ▶ *The internal services*: The following enforcement activities are performed at the service layer:
 - Incoming token validation using the trust service.
 - Message level security validation using the trust infrastructure.
 - Authorize the users to access the service. This is generally achieved using the internal Policy Decision Point.
 - Audit the events.
- ▶ *The external company portal and the external services*: These components may in turn enforce security with their own decision points.

Monitoring and reporting

In this scenario, the monitoring activities are used to ensure that the portal response times are acceptable, either for internal and external users. Disk space, CPU utilization, and network bandwidth utilization may also be controlled to ensure a high quality of service.

Reports can be generated periodically, on a per service basis or to track some specific activities. For example, a company may need to know how many pages a user sees during his visit, or how many times a user comes during a month.

5.2.3 Business Security Services

This section focuses on the deployment environment required for this scenario. Figure 5-17 shows the classification of the management solutions from the IBM SOA Security Reference Model.

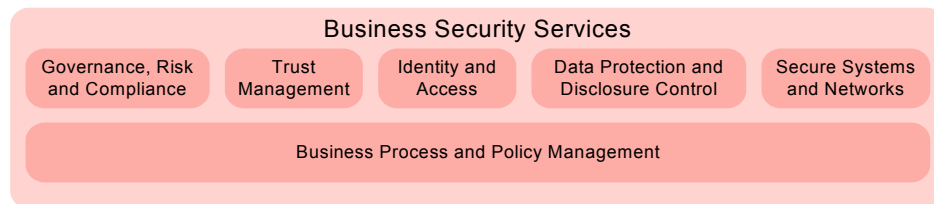


Figure 5-17 Business Security Services from the IBM SOA Security Reference Model

Each of these items is discussed in this section in the scope of this scenario.

Governance, risk, and compliance

As part of the governance activities in this scenario, the company needs to agree with its external partners on the services and on the information exchanged. This may include the type of protocol and assertion used to federate the user accounts as well as the role played within the federation by each company.

Internal governance activities include the definition of the security policies to access the services. A company may decide to expose a first set of core services to a group of partners before expanding the use of these services to others.

As a result of the risk management process, examples of resultant compliance objectives in this scenario may include:

- ▶ Revalidation of the users' accounts from the managers and the system administrators after a certain period of time.

For example, managers may be required to revalidate all their employees' accounts every quarter. This helps in detecting accounts that are not required anymore.

A system administrator (for example, from RACF) may be notified every week of all the accounts that have been created on the system. The administrator can then block the accounts if they do not comply with the company identifier policies.

- ▶ Generating the appropriate reports to comply with the different regulations the company has to deal with.

Trust management

In this scenario, users of the portal need to be made aware of the terms and conditions of use, privacy policy, and so on. This may be done at the overall organizational level for users of the portal, or on an individual user level. This is one of the business aspects of trust management.

At the technology level, trust management may define:

- ▶ Standards for the strength of cryptographic ciphers to be used when accessing the portal via HTTPS
- ▶ Standards for authentication mechanisms and associated metadata, such as password strength
- ▶ Key management for X.509 server certificates for the portal and service components

Identity and access

Identity and access management policies have to be defined as part of the solution deployment.

Identity management

Identities have to be managed through their overall life cycle in this scenario.

Initial identity creation can be obtained from an authoritative source such as the HR System in this scenario. Synchronization policies then allow to propagate any update to this system to the different repositories identified in "Identity foundation" on page 124.

The use of an enterprise provisioning solution provides a single entry point to manage the identities and the life cycle of these identities within the company, based on the security policies. In this scenario the use of such solution can help in applying provisioning policies towards the different repositories. It can significantly improve the user experience as this provisioning ensures a new

employee has access to the company portal and the different services aggregated in this scenario within a short period of time.

Validation processes can be defined so that a manager needs to validate an account creation for any person managed.

Identifier policies are defined so the accounts are created based on the policies of the company. For example, the Windows® account identifiers are formed with the first letter of the first name and the complete family name.

Password policies can be defined to enforce security. Setting complexity and expiration dates to force password changes are common examples of password policies.

Another important aspect of security is revalidation of the accounts. A system administrator may be required to validate all the accounts every month to check their compliance to the security policies of the companies.

Self-service capabilities can be provided to the user, through a Web interface for example. It allows the user to update personal information and to reset his passwords (and possibly propagate a password change from a system to another).

Access management

In this scenario, the appropriate access control policies have to be deployed to the different decision points. An approach can be to externalize as much as possible the authorization policies to the point of contact, another one can be to let the company portal handle some granularity on these decisions.

For existing services, the access management policies are reused in this scenario.

Data protection and disclosure control

Data protection management identifies the resources needing protection and the controls required on those resources. For example:

- ▶ In this scenario, a PKI may be used to authenticate the users accessing the portal from the external network. The cryptographic key stores have to be stored on dedicated hardware for these purposes.
- ▶ The trust infrastructure also uses some cryptographic key stores to store its own certificates as well as partner certificates. The keys from these certificates are then used to digitally sign and encrypt the security tokens or some content of the messages.

In that case, the file system hosting these different key stores has to be protected from external access and needs to be encrypted.

- ▶ Finally, an operating system hardening solution can be used to protect the different machines used in this scenario. For each machine, the different file directories must have the appropriate permissions set.

In this scenario, the company can set a privacy policy to require the user consent before federating his accounts with one partner or with a set of partners identified.

As part of the disclosure control in this scenario, the user may have access through its portal to a service aggregating its personal information to make the appropriate changes. This can include personal changes (such as phone number or an e-mail address) and also business information changes (for example, the user may decide that its e-mail address must not be given to any external partner).

Secure systems and networks

Figure 5-18 shows the logical architecture that can be used for this scenario. Firewalls can be used to filter the access from one zone to another.

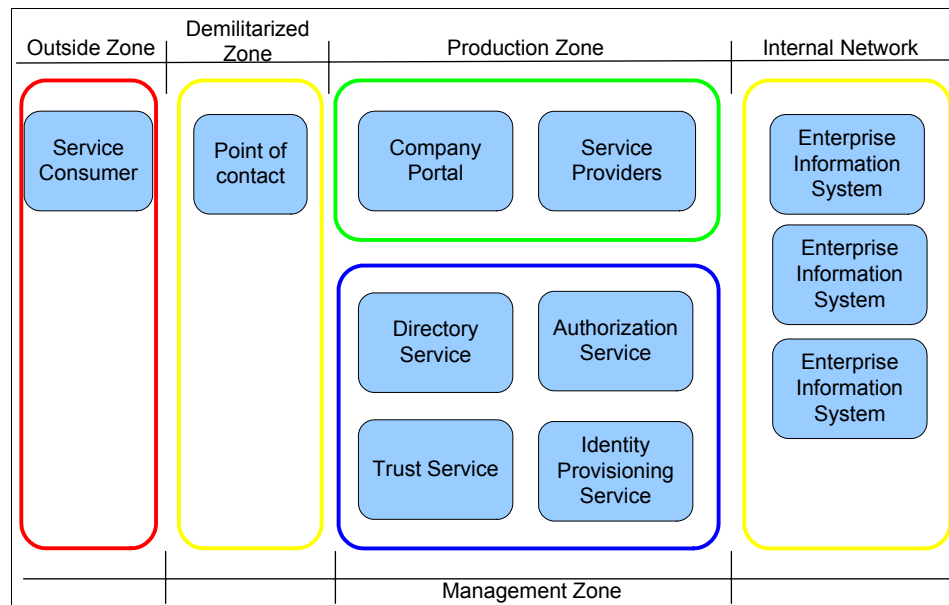


Figure 5-18 Typical logical network diagram for the SOA Foundation Service Aggregation scenario

The service consumer can be any external application calling services exposed through the company portal. Several service providers are defined in the

production zone and each of them may access a different enterprise information system.

A dedicated hardware SSL accelerator solution can also be used in front of the demilitarized zone as part of the solution to encrypt / decrypt any flows going through the network.

Business Process and Policy Management

In the Service Aggregation scenario, some aspects of Business Process and Policy Management include:

- ▶ Managing compliance activities related to authorizing access to the portal based on periodic revalidation of the business need.
- ▶ Establishing policies for access to portal resources based on the incoming user's organization and role.
- ▶ Managing the password expiration policies in the point-of-contact or portal to reduce the risk of exposure due to compromised passwords.
- ▶ Classifying aggregated applications and data so that appropriate data protection and disclosure control can be implemented.

5.3 Summary

This chapter proposed an application of the IBM SOA Security Reference Model to the Service Aggregation scenario. It deliberately focused on the company security requirements and gave an example of a partner integration. It can be extended to the integration of other partners and can be used as a reference to apply security within a context of a SOA scenario.



Part 3

Securing the Service Creation scenario

Part 3 provides an end-to-end working example representing the direct exposure of existing CICS applications as services and securing the exposed service realization example for the IBM SOA Foundation Service Creation scenario.



Business scenario

In this chapter, we describe the business context and the requirements for our sample implementation, a scenario that involves two hypothetical corporations.

6.1 Business model

Descriptions of the initial context of the ITSOBank and ITSOTelco organizations are provided here, along with insight on the key business drivers and IT challenges the organizations are facing.

6.1.1 Overview

The overview explains the background of the example scenario and gives a short description of the companies involved.

The corporations involved in the scenario are:

- ▶ ITSOTelco

A large telecommunications company servicing both individuals (retail customers) and corporate customers. In an effort to provide a rich user experience, ITSOTelco has partnered with its corporate customer and service provider, ITSOBank, to leverage federation technologies. Through effective application of these technologies, ITSOTelco is intending to deliver seamless interactions for its customers using browsers and mobile devices.

- ▶ ITSOBank

A progressive retail banking corporation looking for new ways to serve its customers. ITSOBank has partnered with ITSOTelco to provide a new service to users who are ITSOBank employees. These employees will be able to access their account information at ITSOBank from ITSOTelco's customer portal. They can check their account information at ITSOTelco without having to separately log in to the ITSOBank application.

Figure 6-1 on page 153 shows the overall configuration. We further describe the detailed configuration and technical aspects of securing the services between the two companies in the following sections.

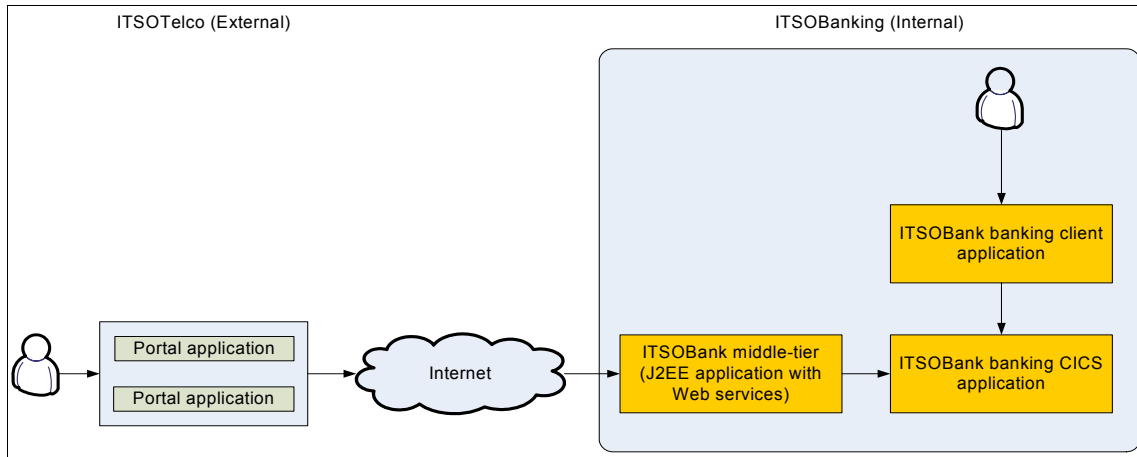


Figure 6-1 Overall scenario logical architecture

Figure 6-2 outlines the required security mechanisms to allow authenticated access to the Web service and to achieve single sign-on.

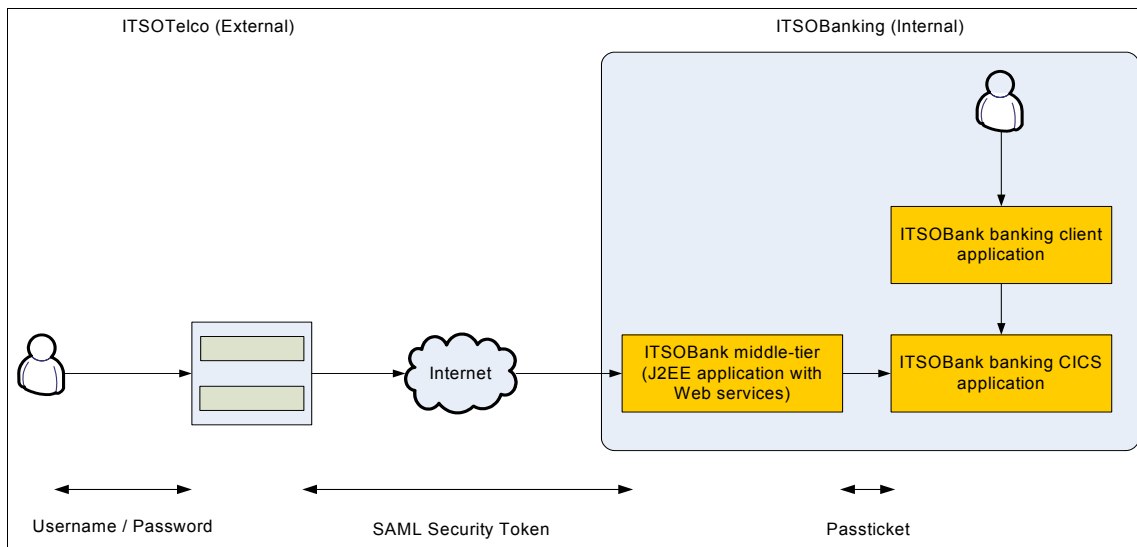


Figure 6-2 Security mechanisms

6.1.2 Initial context - ITSTelco

ITSTelco has developed a Portal Solution to offer services to corporate customers. ITSOBank is one of their corporate customers. ITSOBank employees are also customers of ITSTelco.

Note: The implementation and deployment of the ITSOTelco Portal solution is considered *in place*. For further information how to deploy a secure portal solution, please refer to the IBM Redbook *Develop and Deploy a Secure Portal Solution Using WebSphere Portal V5 and Tivoli Access Manager V5.1*, SG24-6325.

Our example scenario focuses on the ITSOBank implementation.

The first service that ITSOTelco wants to offer allows access to the current balance of their bank accounts in the portal. At a later date, other services are planned to be consumed by the portal to provide more functionality for the ITSOBank employees.

6.1.3 Initial context - ITSOBank

Figure 6-3 depicts the initial IT context for ITSOBank. The IT infrastructure shown in the figure outlines the way to access the application via a CICS client, for example, a 3270 screen. The underlying security for the CICS application is provided by RACF. Services and components that are not vital to understand the scenario have been omitted for clarity.

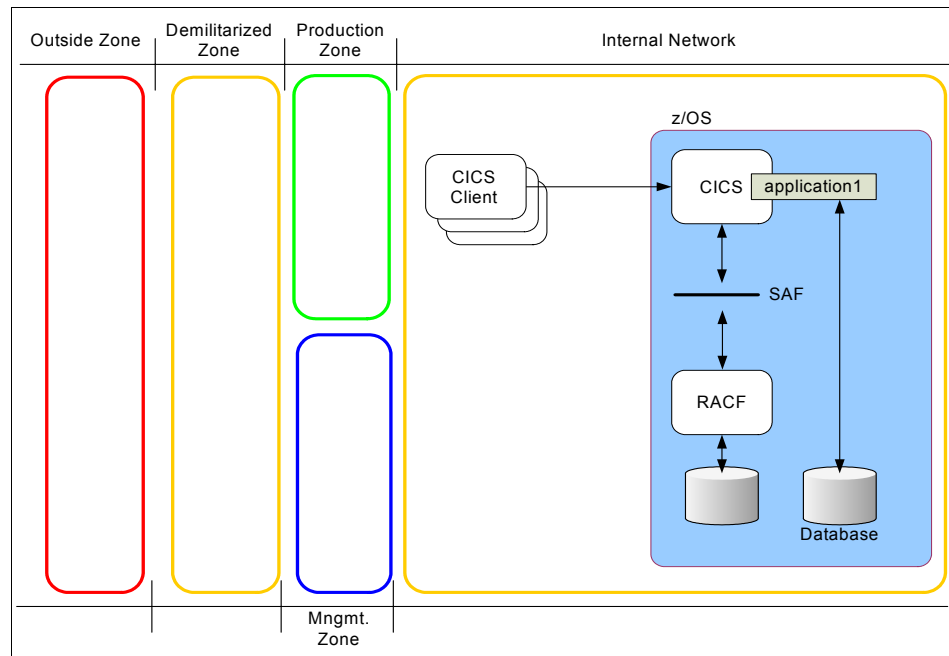


Figure 6-3 ITSOBank initial I/T context

6.1.4 Preliminary SOA engagement

ITSOBank has performed a series of workshops with the executive management team and major stakeholders to gain a better understanding of the company's business objectives and IT challenges. ITSOBank views SOA as a strategic initiative they would like to pursue for both integration and application development at a larger enterprise level. ITSOBank desires a transition to SOA that can leverage existing applications and infrastructure.

To facilitate the adoption of SOA, the company wishes to establish early proof points in phases.

- ▶ Phase 1: Expose the *query balance* functionality via a service component.

In Phase 1, a specific function, query balance, of the *COMMAREA COBOL* banking application hosted by the CICS Transaction Server will be exposed indirectly by creating a middle-tier Web service to access CICS. This middle-tier Web service wraps the EJB™ that uses the *CICS ECI resource adapter*¹ to connect to the CICS Transaction Gateway to access the CICS Transaction Server. This is shown in Figure 6-4 on page 156.

¹ The CICS ECI resource adapter used in this scenario is a JCA adapter for WebSphere Application Server shipped with the CICS Transaction Gateway.

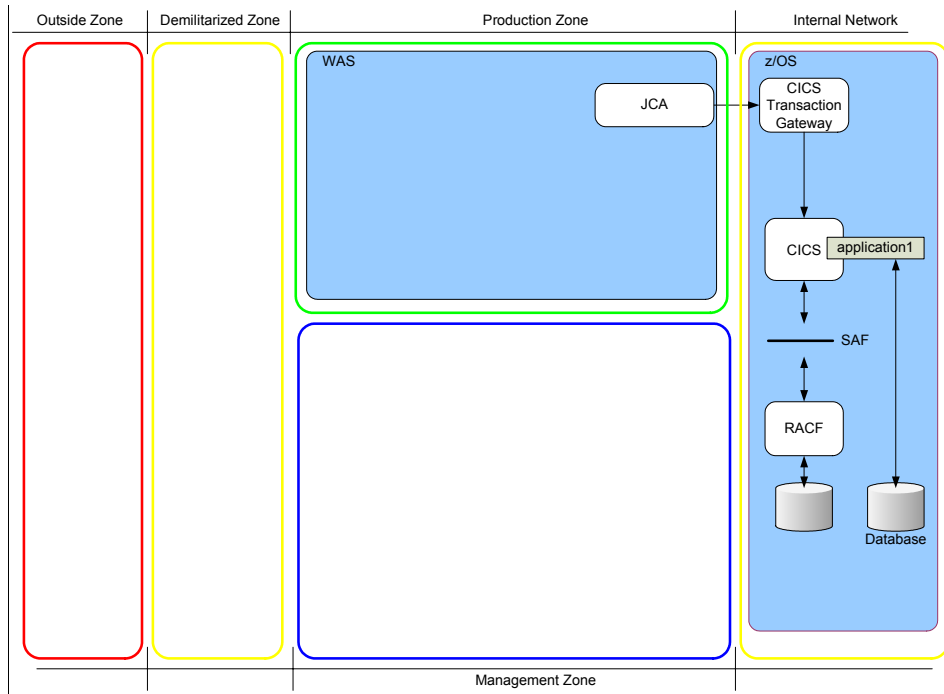


Figure 6-4 Using the CICS Transaction Gateway to access CICS from WebSphere Application Server

- Phase 2: Secure the Web service by applying the SOA Security Reference Architecture.

In the next sections, we discuss parts of the business logic of the initial context and the current security infrastructure.

6.1.5 Business logic

The business logic of retrieving the current balance of an account is implemented in a CICS application. As the service will be indirectly exposed, the middle-tier will only be used to expose the services from the CICS application and will contain no business logic.

In order to accomplish Phase 1 of the project, ITSOBank has developed a simple online banking application that runs on WebSphere Application Server. The solution is currently in its first release and is only available internally within the bank for access by developers. As such, the application presently has limited functionality and limited security. It provides a single Web service to query the balance of an account by calling the corresponding function in the CICS application.

Figure 6-5 shows the architecture after the deployment of the Web application. As the application is currently only deployed in the intranet, system component placement is less complex than it will be when the solution becomes available as a secure application accessible through the Internet.

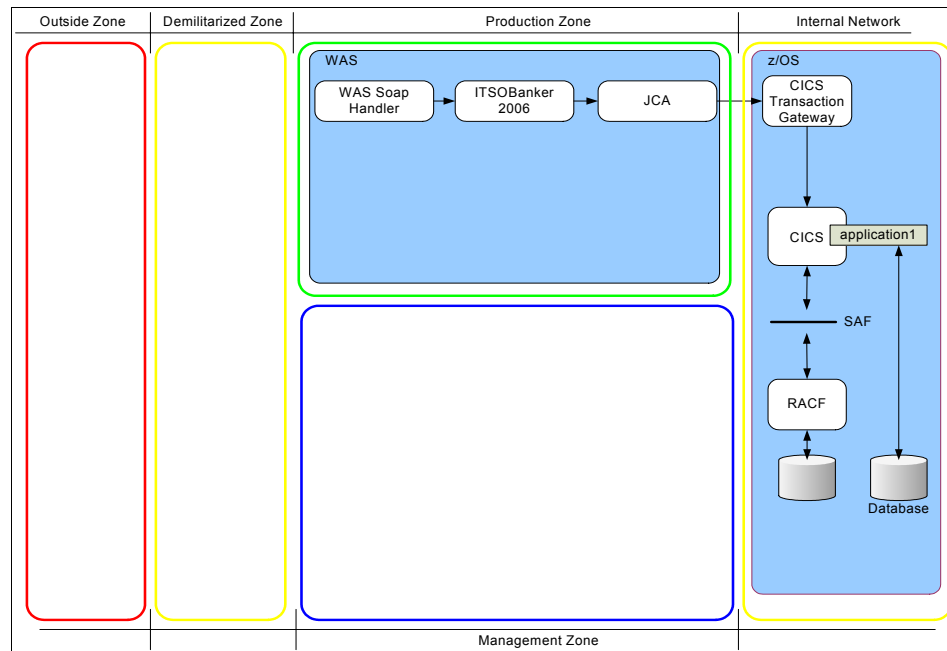


Figure 6-5 ITSOBank application

6.1.6 Authentication and authorization

Under z/OS the concept of pleadable security is important; it means that any security product or component can be plugged into the z/OS operating system. This is possible through the System Authorization Facility (SAF) available under z/OS. As shown in Figure 6-6, at ITSOBank, security in the CICS environment is handled by RACF. As a consequence, every employee of ITSOBank has a RACF account that allows them to work with the CICS application.

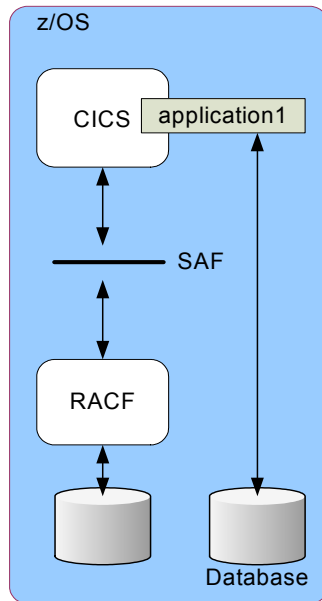


Figure 6-6 Using RACF with CICS

An explanation of the basic concepts of how security is handled in the CICS environment using SAF and RACF can be found in “Identity Attribute Service (IdAS)” on page 384.

6.2 Business requirements

This section describes the business requirements for the ITSOBank scenario. These requirements have been derived from the principal users of the application.

- ▶ BR1: Reuse existing application functionality.

The purpose of this initiative is not to implement new applications, but rather to reuse existing applications in new ways.

- ▶ BR2: Make the services easy to consume.
The goal is to drive rapid adoption of the services being offered to business partners, so the effort to consume these services should be low.
- ▶ BR3: Services should be secure.
Secure access to information, especially beyond the corporate boundary, is critical.
- ▶ BR4: Auditing should meet regulatory requirements.
Linked with BR3, audit trails should be able to identify the individual user that was performing a transaction at every point in the transaction.
- ▶ BR5: The system should be designed for growth.
If successful, use of the solution can grow rapidly. The design of the solution should support this.
- ▶ BR6: The system should be available as close to 24x7 as is practical.
Reasonable availability is required during the proof-of-technology phase, but availability of a production environment should be designed to be 99.9% or better.

6.3 Technical requirements

This section describes the technical requirements that support the business requirements in the previous section. They are broken up into functional and non-functional requirements.

Note: Security requirements, usually described within the non-functional requirements of an overall enterprise architecture, are given their own section to highlight the security focus of this IBM Redbook.

6.3.1 Security requirements

This set of technical requirements support business requirements BR3 and BR4. As is usually the case, the technical requirements draw out many more facets than the often innocuous business requirements.

▶ SR1: Authentication to the Web service

Whether the route to the Web service originates within the ITSObank organization or from a IBM Business Partner location, users accessing the service should be positively identified. No reliance should be placed on system or application identities. For example, it should be known that it is user *Joe* accessing the Web service, not *the portal application at ITSOTelco*.

▶ SR2: Authentication to CICS Transaction Gateway

Users must authenticate to RACF using their own user identity, and not a shared service/application login. The user's RACF password should only be stored in RACF itself, and not in any intermediate identity stores.

▶ SR3: Identity mapping

The user's external user ID and their RACF user ID may not always be the same, but a one-to-one mapping between them has to be possible.

▶ SR4: Authorization

Access to the services is only granted to resource consumers that have been granted authority to use them.

▶ SR5: Centralized policy management

All security policies should be stored in a centralized policy store for ease of management and demonstration of compliance.

▶ SR6: Transport level security

The communication channel between service consumers outside the ITSObank enterprise over the (untrusted) Internet must be encrypted using transport level security, such as SSL/TLS.

▶ SR7: Message integrity

In order to assure the integrity of communications, the content of Web service messages (requests and responses) must be signed.

▶ SR8: Message confidentiality

In order to guarantee the confidentiality of communications, the content of Web service messages (requests and responses) must be encrypted.

▶ SR9: DMZ termination of inbound requests from partners

For Web service requests originating outside of ITSObank, a termination point is required in the DMZ before requests are proxied to the network where

the Web service is deployed. At a minimum, the DMZ termination point should perform content validation, authentication, coarse-grained authorization, and audit.

▶ SR10: Auditing

Each component of the solution has to be enabled for auditing. A mechanism has to be available to submit, persistently store, and report on audit data submitted as events.

▶ SR11: Use of standards

The solution should employ applicable open standards for security management in SOA environments. This assures maximum interoperability with minimal customization.

▶ SR12: Account recertification

The accounts of every employee have to be recertified every three months.

▶ SR13: Security token standards

The SOA Governance board has provided a guideline for use of signed SAML 1.1 assertions for identity assertions from external service consumers. They have also provided a guideline to use SAML 2.0 security tokens for identity assertions within the enterprise.

6.3.2 Other functional requirements

This section describes the other functional requirements that have been identified.

▶ FR1: Expose CICS transactions as a Web service.

Service-oriented architecture supports the business and technical objectives of this project.

▶ FR2: Create sample Web service client for internal use.

Enable internal applications to use the services. This provides an initial delivery and test scenario before exposing the applications to business partners.

▶ FR3: Expose selected Web services to an external partner.

Expose selected Web services to external partners to enable them to use the services of the banking application. ITSOTelco is the first potential consumer of this service.

▶ FR4: Create a template Web services client application to demonstrate to external partners how to consume these services.

Create a template Web services client to be used by external partners to consume the externally exposed services of the ITSObank application.

6.3.3 Other non-functional requirements

This section describes the non-functional requirements that have been identified. They cover a broad range of themes, such as:

- ▶ Performance
- ▶ Scalability
- ▶ Availability
- ▶ Maintainability
- ▶ Monitoring

These non-functional requirements define the design and development of the operational environment of the system. They also define the way the system is configured and managed.

Important: This example primarily focuses on the security requirements so the non-functional requirements are just briefly mentioned.

- ▶ NFR1: The response time of this service should be no more than three seconds.
Feedback from business partners indicate this is an acceptable latency for integration with their customer portals.
- ▶ NFR2: 50% of the ITSOPBank employees will use the ITSOTelco portal per day.
This leads to a usage of the Web service of 10,000 times a day.
- ▶ NFR3: No server in the solution should be more than 60% utilized.
This allows for future growth.
- ▶ NFR4: The overall architecture should be able to achieve availability close to 99.9%.
It is recognized that the first phase is a proof-of-technology, and that availability requirements are less critical at this time. Availability requirements will likely change as services become more critical to the business.
- ▶ NFR5: The maintenance of the security component of the system must have a reasonable and predictable cost.

This concludes the description of the business context for our customer scenario.



Solution design

In this chapter, we introduce the solution design for our scenario covering the following aspects:

- ▶ Solution architecture introduction
- ▶ IT Security Services
- ▶ Security Policy Infrastructure
- ▶ Business Security Services

7.1 Solution architecture introduction

After analyzing the business and security requirements outlined in Chapter 6, “Business scenario” on page 151, we now design the secure solution. We will apply the IBM SOA Security Reference Model to the existing architecture that indirectly exposes the existing CICS application. According to the IBM SOA Security Reference Model, the solution architecture contains three main components:

- ▶ IT Security Services
- ▶ Security Policy Infrastructure
- ▶ Business Security Services

Figure 7-1 shows the Indirect Exposure Architectural Pattern of the SOA Service Creation scenario.

Note: Please refer to 3.1.2, “Indirect exposure architectural pattern” on page 65 for a more detailed description on the Indirect Exposure Architectural Pattern.

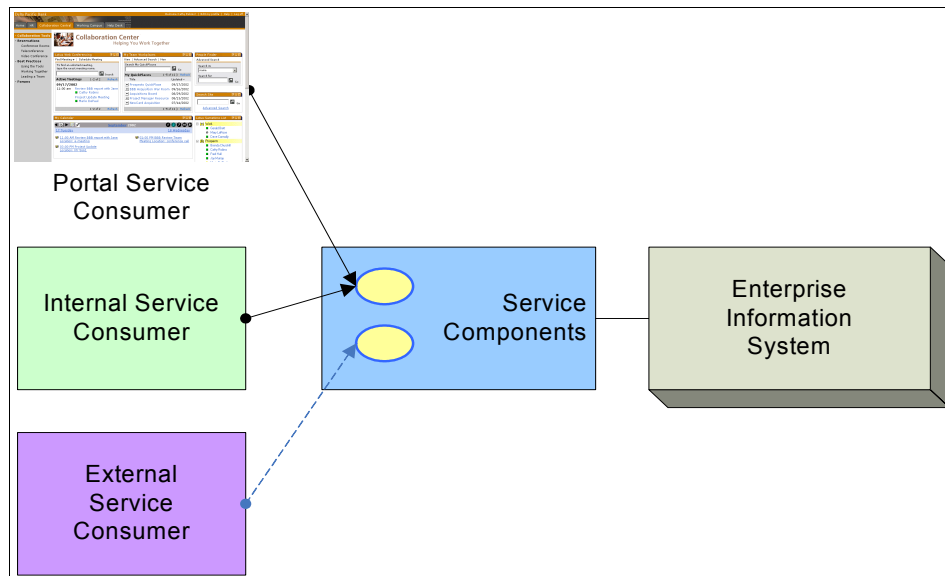


Figure 7-1 The Indirect Exposure Architectural Pattern

In the next section we discuss how the security services shown in Figure 7-2 on page 165 can be applied to the ITSOBank scenario by discussing all

components of the security services and how they match the requirements from ITSOBank. By adding the different components step-by-step to the architecture shown in the initial context description, the diagram will evolve to the secured solution architecture.

Note: A detailed discussion about applying the IBM SOA Security Reference Model can be found in Chapter 3, “IBM SOA Foundation Service Creation scenario” on page 61.

Figure 7-2 shows a sample architecture to meet the requirements laid out in Chapter 6, “Business scenario” on page 151 and how the SOA Security Reference Model is applied to the architecture. In this example, a DataPower appliance is serving the purpose of both an *XML firewall* and *Web services gateway*. It acts as a proxy and provides the first layer of defense in depth strategy for the whole solution. The WebSphere Application Server hosts the service that leverages the information accessible from CICS.

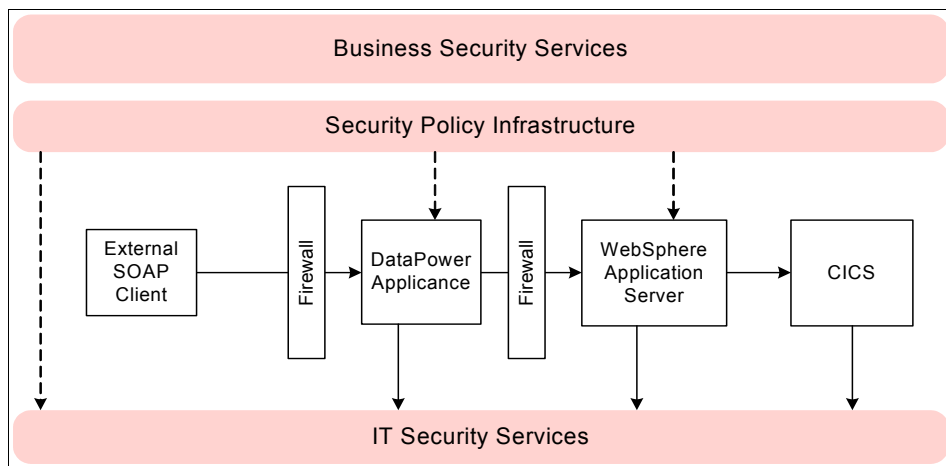


Figure 7-2 Applying the SOA Security Reference Model

7.2 IT Security Services

In reference to the checklist introduced in 3.2.1, “IT Security Services” on page 67, the marked services will be considered in the ITSOBank scenario. Table 7-1 is also used as a starting point for the analyses and design phase.

Table 7-1 A checklist of the IT Security Services from the IBM SOA Security Reference Model that can be applied to the ITSOBank scenario

Security Services	Application to the Service Creation Scenario	Evaluated
Identity Services	Identity foundation	X
	Identity provisioning	X
	Identity federation	X
Authentication Services	Service consumer (user or system)	X
	Service components	X
	Enterprise Information System	X
Authorization and Privacy Services	Service components (authorization)	X
	Enterprise Information System (authorization)	X
	Service components (privacy)	
	Enterprise Information System (privacy)	
Confidentiality and Integrity Services	Service consumer to service components	X
	Service components to EIS	X
	EIS (data protection)	
Audit Services	Identity (foundation)	X
	Identity (provisioning)	X
	Identity (federation)	X
	Authentication (consumer)	X
	Authentication (service components)	X

Security Services	Application to the Service Creation Scenario	Evaluated
	Authentication (EIS)	X
	Authorization (service components)	X
	Authorization (EIS)	
	Privacy (service components)	
	Privacy (EIS)	
	Message protection (service consumer to service component)	X
	Message protection (service component to EIS)	X
	Data protection (EIS)	

7.2.1 Identity Services

The identity services described in “Identity Services” on page 68 are composed of the following components:

- ▶ Identity foundation
- ▶ Identity provisioning
- ▶ Identity federation

Identity foundation

There are two user repositories that are used in the solution: a directory server used by the middle-tier (in our example, WebSphere Application Server) and the RACF database, which contains the identities of registered users of the CICS banking application. The directory server needs to be deployed; the RACF database is already implemented.

Figure 7-3 shows that an *LDAP user registry* is added to the solution architecture. It stores common identity information for the enterprise. In our scenario, we deploy IBM Tivoli Directory Server (TDS in the picture). The RACF identity database is also shown, as it is part of the z/OS System.

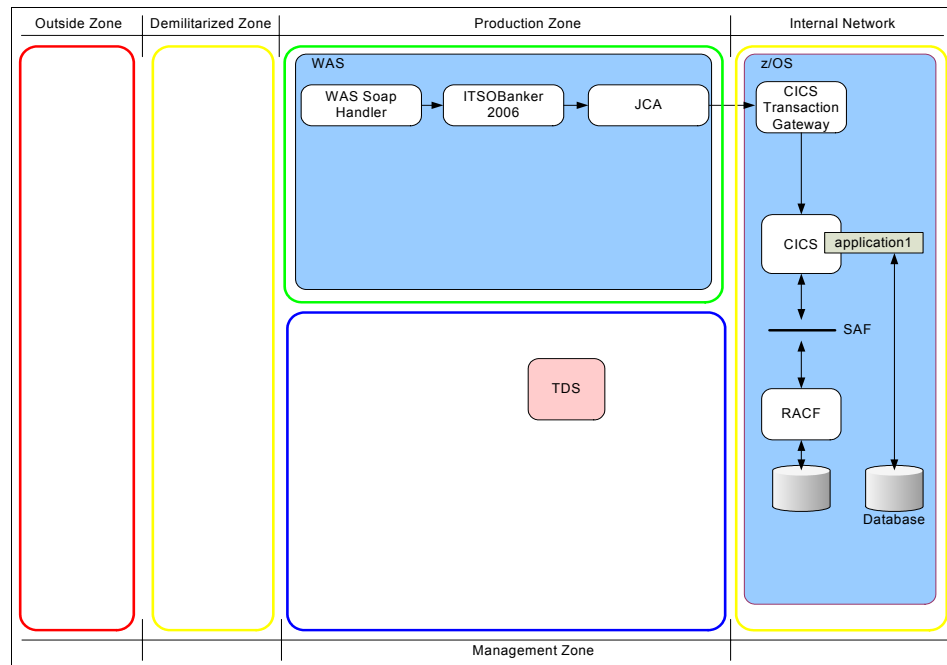


Figure 7-3 Solution architecture / User registries

Identity provisioning

In order to add, delete, or modify individual account information, an identity provisioning solution needs to be implemented. IBM Tivoli Identity Manager provides a secure, automated, and policy-based user life cycle management solution. It guarantees that only the entitled identities with the correct attributes are provisioned to the LDAP directory and the RACF database.

Identity Manager ensures that the RACF user ID of each employee in ITSOBank is synchronized as an attribute of the user object in the LDAP registry. This attribute is used later when mapping between the user ID used on the Web service call and the RACF user ID has to be done.

Figure 7-4 on page 169 shows the architecture including Identity Manager (TIM in the picture) with three adapters. One adapter for the IBM Tivoli Directory Server, one adapter to RACF, and one adapter to IBM Tivoli Access Manager for e-business (which is introduced in 7.2.2, “Authentication and authorization services” on page 170) (TAM in the picture).

In order to perform authentication and authorization in WebSphere Application Server, *global security* is activated and the Tivoli Directory Server is configured as the user repository.

Tip: For more information about how to configure security in WebSphere Application Server, we recommend the IBM Redbook *IBM WebSphere V6.0 Security Handbook*, SG24-6316.

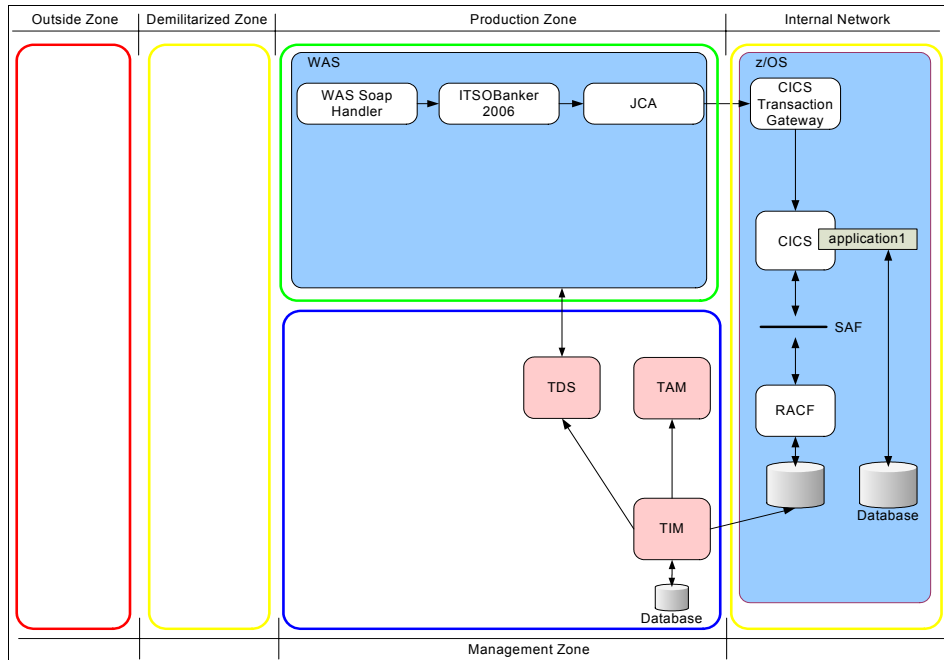


Figure 7-4 Solution architecture / Identity provisioning

Identity federation

For this solution design, identity federation deals with token mediation and identity mapping, which is provided by a standard component called *Secure Token Service* (STS). STS is an implementation of the WS-Trust specification and has the capabilities to validate an inbound token and issue a new token based on policies. This is discussed more in the following section, as it is integrated in the authentication and authorization workflow.

7.2.2 Authentication and authorization services

Authentication and authorization for Web service requests in our scenario is handled by calling the Security Token Service (STS) of IBM Tivoli Federated Identity Manager. The STS manages the handling of security tokens that are used to authenticate and authorize the Web service requests. Federated Identity Manager itself is configured to use Access Manager for e-business to perform authentication and authorization. To transmit the relevant information from the point of contact to the STS, the request is structured according to the WS-Trust specification that defines a standardized format for security token requests. To determine how to process the request, the STS uses defined fields such as *Issuer* and *AppliesTo*. To process the tokens, the STS uses modules, module instances, and trust service chains.

For each type of token, the STS has a security token *module* that handles the trust relationship. These modules are responsible for creating tokens, validating tokens, and exchanging token types.

Trust modules perform specific functions based on the mode in which they operate. In order to execute the complete and correct sequence of necessary functions, the STS configures *trust module instances* into *trust service chains*. Trust service chains are groups of module instances that are configured to be used together.

The trust service chain is configured to use Access Manager for e-business as an authentication and authorization service. Access Manager for e-business authenticates the presented user ID and authorizes the request to invoke the Web service.

As the security context for a user has to traverse different parts of our solution, we use the STS to transform security tokens and to map the identities so they match the identities in the respective registry.

Figure 7-5 on page 171 shows two different ways to invoke the ITSOBank Web service:

- ▶ An external client invoking the ITSOBank Web service from outside the ITSOBank intranet. The DataPower appliance, used as a first line of defense, has to authenticate and authorize these requests.
- ▶ An internal client invoking the ITSOBank Web service from the intranet.

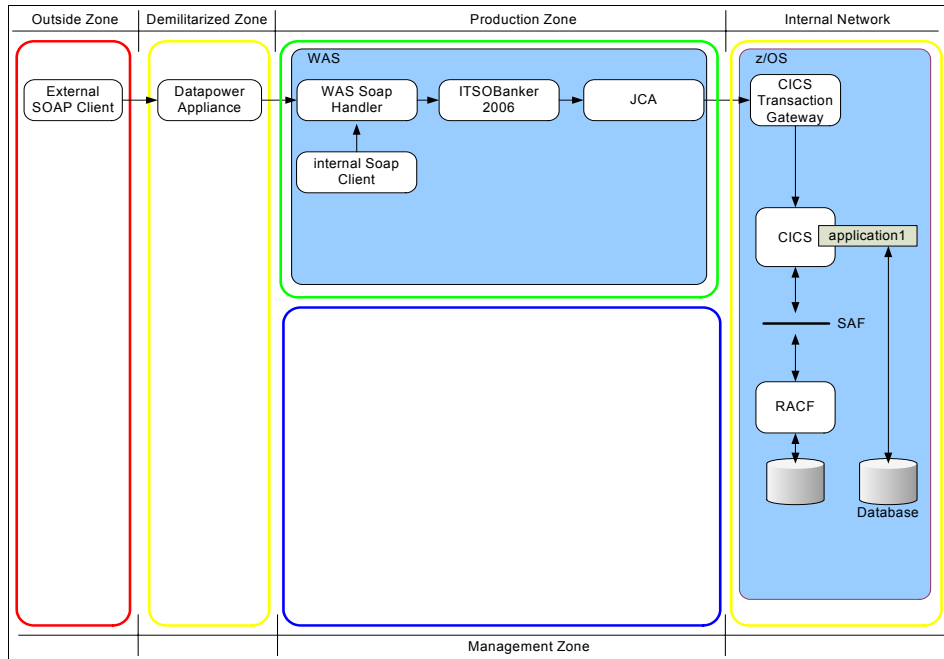


Figure 7-5 Internal and external SOAP clients

This leads to the following system boundaries where security tokens are exchanged and trust service chains have to be configured.

- External SOAP client to DataPower appliance

When calling the ITSOBank service, a SAML 1.1 security token is included in the Web service request by the ITSOTelco portal solution. The ITSOTelco portal generates this security token based on the authenticated identity for the user session in the portal. The DataPower validates WS-Security based message protection and then extracts this token and sends a the SAML1.1 token to the STS. If the token is valid and the user can be authenticated and authorized, the STS issues a new SAML 2.0 security token and returns the *RequestSecurityTokenResponse* to the DataPower appliance.

Figure 7-6 shows the system boundaries where security tokens are exchanged and trust service chains have to be configured when an external client invokes the ITSOBank Web service.

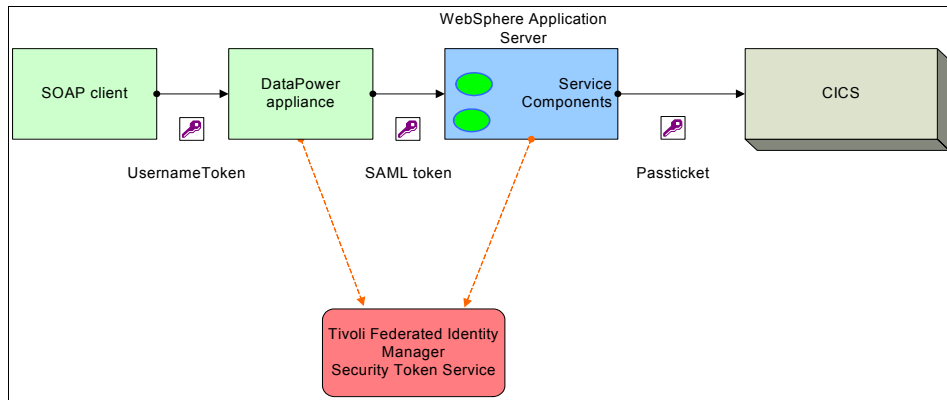


Figure 7-6 Token exchange / external client

- ▶ DataPower appliance to WebSphere Application Server
The SAML 2.0 token issued by the STS is sent to the WebSphere Application Server, which authenticates and authorizes the request by using the WSSM extension and the STS.
- ▶ Internal application client to WebSphere Application Server
The process of authenticating and authorizing an internal application client is handled the same way as authenticating and authorizing incoming Web service requests from the DataPower appliance. It only differs in the security token profile. The security token profile used to authenticate an internal application client is a user name token. Figure 7-7 on page 173 shows the system boundaries where security tokens are exchanged and trust service chains have to be configured when an internal client invokes the ITSOBank Web service.

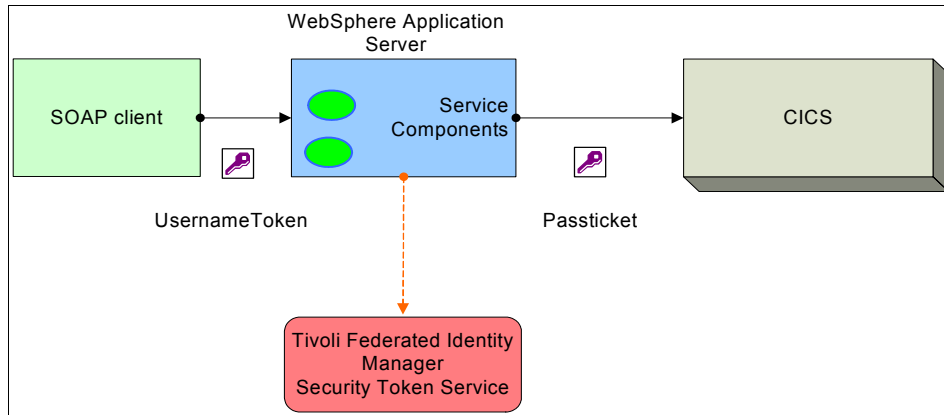


Figure 7-7 Token exchange / internal client

► WebSphere Application Server to CICS Transaction Gateway

To connect to the CICS Transaction Gateway, we need to first map the authenticated user ID within WebSphere to the RACF user ID. As we configure Identity Manager to store the RACF ID of each employee as an attribute in the LDAP, the mapping mechanism uses this attribute to obtain the RACF ID and generate a Passticket. An appropriate trust chain to implement this functionality has to be in place.

Authenticating and authorizing external requests

The SOAP client supplies a SAML 1.1 security token to identify the requester. The DataPower appliance is configured to call the Federated Identity Manager STS to:

- Validate the token.
- Map the identity of the consumer to the matching user ID in the enterprise registry. As ITSOTelco submits the employee's internal user ID, a one-to-one mapping is configured.
- Authorize the request.
- Generate a SAML 2.0 security token for further processing.

Figure 7-8 shows the sequence of the trust chain and which modules are configured into the trust chain.

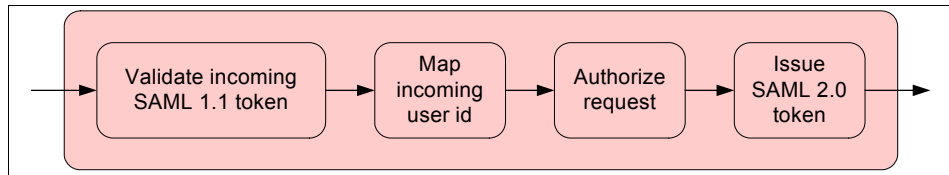


Figure 7-8 Trust chain - Authenticating and authorizing external requests

The components used to authenticate and authorize the incoming requests are shown in Figure 7-9. The Federated Identity Manager (FIM in the picture) STS uses Access Manager for e-business to authenticate the user and authorize the request for the requested resources.

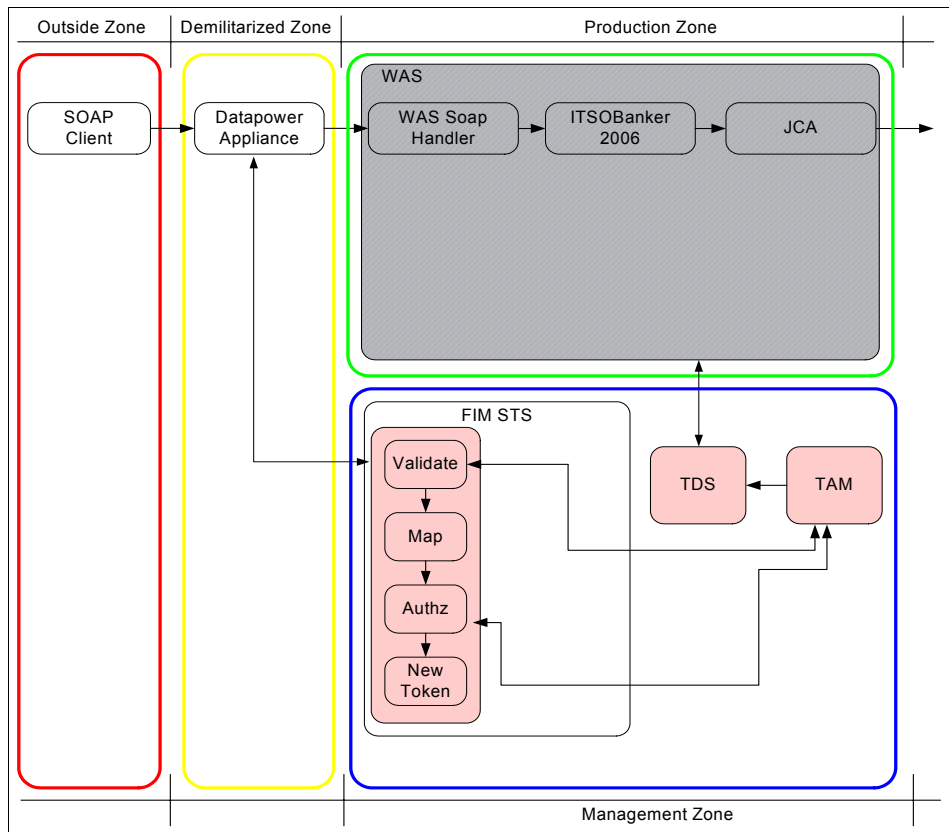


Figure 7-9 Components - Authenticating and authorizing external requests

Authentication and authorization to WebSphere

Incoming requests from either the DataPower appliance or from internal applications are authenticated and authorized by calling the STS. Incoming requests from the DataPower appliance are carrying a SAML 2.0 token in the message while requests from internal clients authenticate with a user name token. So we have to consider both cases in the design of the trust service chains.

To handle WS-Security token processing within WebSphere Application Server, the Web Services Security Management (WSSM) extension shipped with Federated Identity Manager is used. The WSSM components are:

- ▶ Token consumer
- ▶ Login modules
- ▶ WSSM token generator
- ▶ Callback handlers

The *token consumer* receives the tokens so they can be validated, exchanged, or evaluated as a part of an authorization check. The *WSSM token generator* is called as a part of the WS-Security authentication process of WebSphere Application Server when a Web service request message is received.

The token consumer in our scenario is configured to call the STS. It creates a *SecurityTokenRequest* message and sends it to the STS. The STS processes the request and returns a *SecurityTokenResponse* message. The token consumer then sends the token from the response message to the *callback handler*, which makes the token available for the appropriate *login module*. The token then is used to log in and the Web service is invoked.

The login modules are Java™ Authentication and Authorization (JAAS) login modules. These login modules use the credentials within the security tokens to perform a login. In our scenario, the SAML 2.0 login module is used.

The trust chain depicted in Figure 7-10 shows the appropriate sequence. This trust chain requires a SAML token or a user name token as inbound security token. After authenticating and authorizing the request, a SAML token is issued.

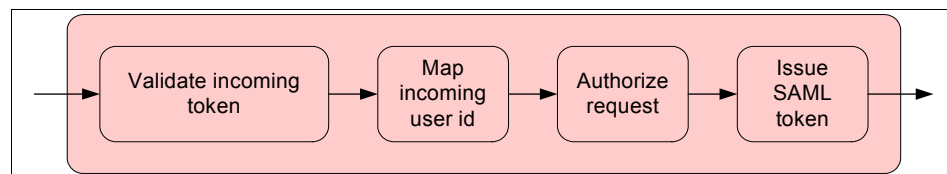


Figure 7-10 Trust chain - Authentication & Authorization to WebSphere

This SAML 2.0 token is used to create an JAAS Principal by invoking the SAML assertion login module after the trust chain is passed through.

Figure 7-11 shows all the components needed to authenticate and authorize the incoming request. The different communication channels are also shown to clarify which components are called by the individual components.

In the case that an internal application client invokes the Web service, the Federated Identity Manager STS also uses Access Manager for e-business to authenticate the incoming user and authorize the request for the requested resource.

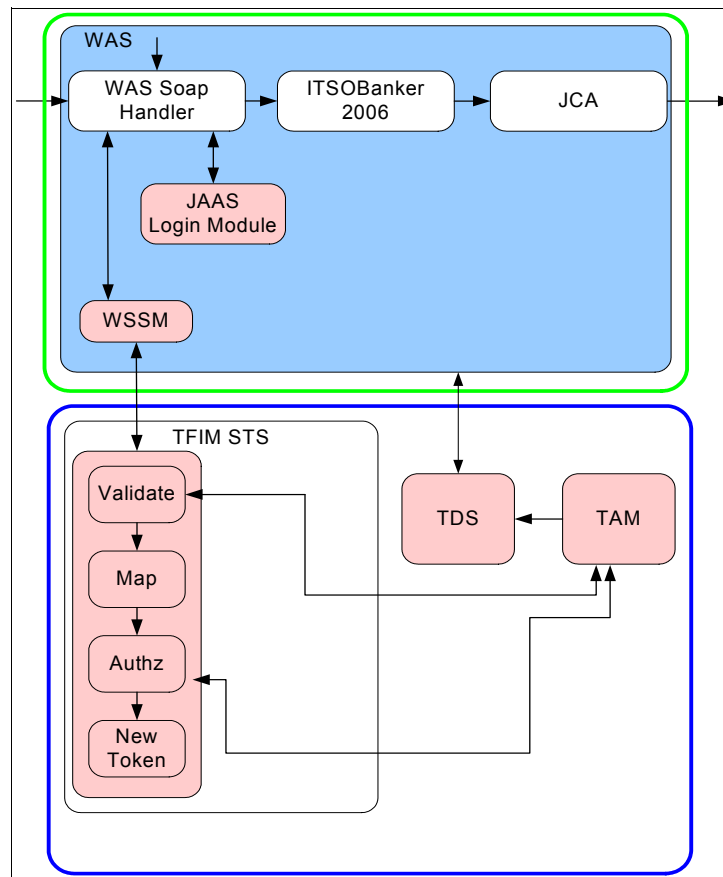


Figure 7-11 Components - Authentication and authorization to WebSphere

Authentication to CICS Transaction Gateway

To sign on to the CICS Transaction Gateway, the RACF user ID and password are needed. To obtain the RACF ID, a special attribute in the LDAP belonging to the signed in user is retrieved. The RACF password is not stored outside RACF (see security requirement SR2 in 6.3.1, “Security requirements” on page 160). A mechanism to deal with that requirement is to generate a *Passticket*, which acts like a one-time password. By performing a JAAS login, which uses a WS-Trust client to call the Federated Identity Manager trust server, we are able to map the user ID to the RACF user ID and generate a Passticket.

The following trust chain in Figure 7-12 shows the configured modules and the sequence.

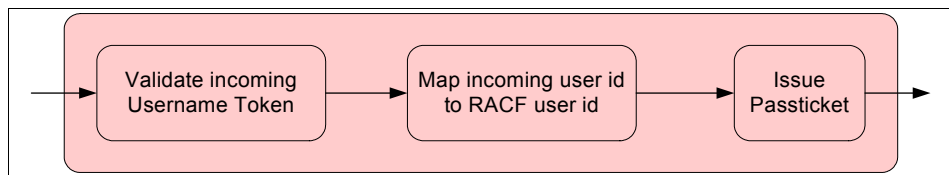


Figure 7-12 Trust chain - Authenticating to CICS Transaction Gateway

The diagram in Figure 7-13 shows the interaction between the components as far as they are identified in this chapter.

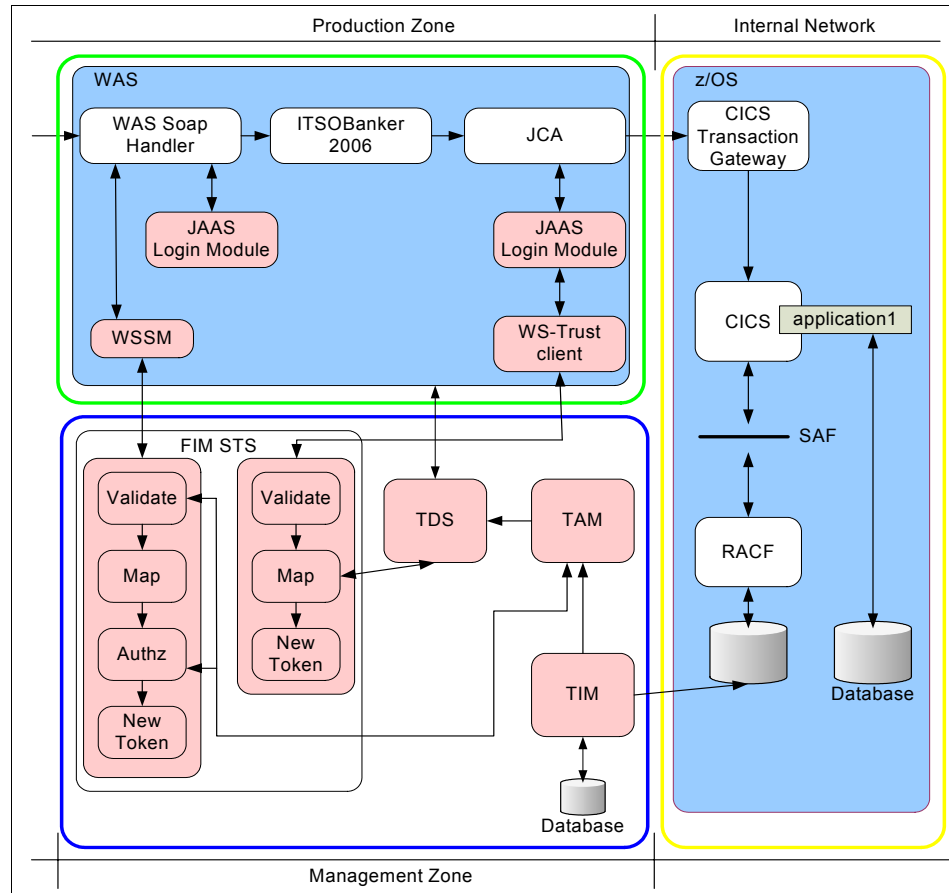


Figure 7-13 Components - Authenticating to CICS Transaction Gateway

Authorizing access to J2EE resources

The ITSOBank banking application implements the J2EE role-based authorization model to perform authorization checks within the application. To request authorization decisions when a J2EE resource is accessed, WebSphere Application Server can use any third-party authorization provider implementing the *JACC provider*. The *Java Authorization Contract for Containers (JACC)* defines a contract between Java 2 Platform, Enterprise Edition (J2EE) containers and authorization providers. Using the JACC provider of Access Manager for e-business, the authorization decisions within the application are delegated to Access Manager for e-business. The communication between WebSphere Application Server and Access Manager for e-business is shown in Figure 7-14.

Using the JACC provider of Access Manager for e-business also offers the functionality to centrally store and manage the policies because the JACC provider communicates with the Access Manager Policy Server to persist the J2EE security configuration data. It also allows you to integrate with the audit subsystem used by Access Manager to audit the authorization decisions made by Access Manager for e-business.

The identity services along with the authentication and authorization services, are addressing the security requirements SR1, SR2, SR3, SR4, SR9, SR11, and SR13.

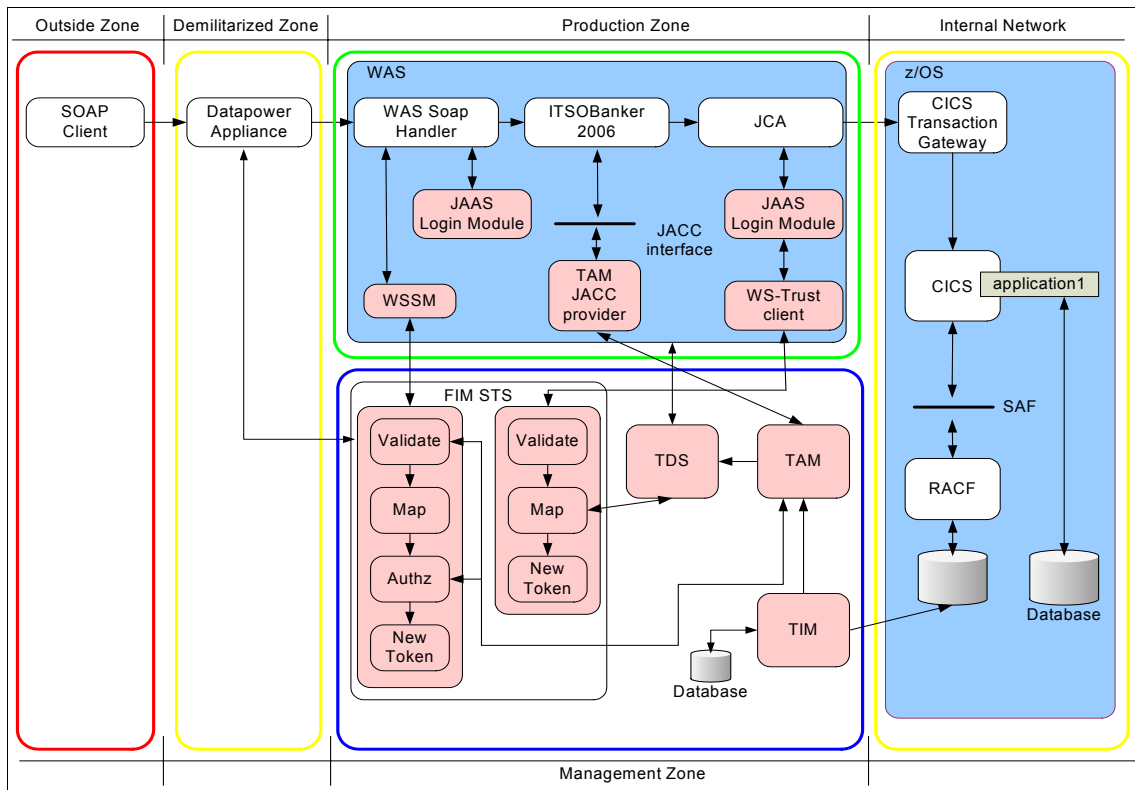


Figure 7-14 Components - JACC Provider

7.2.3 Confidentiality and integrity services

In this section, we examine the requirements for data and message protection.

Data Protection Services

Data Protection Services are concerned with data at rest. In the ITSOBank scenario, the Data Protection Services protect the cryptographic keys stored in the keystores, security configuration files, and the data stored in the database used by the CICS application and the audit data submitted by individual components amongst others.

IBM Tivoli Access Manager for Operating Systems is used to protect the file systems that contain cryptographic keys and configuration files.

Message Protection Services

Message Protection Services are used to protect the data in transit from being:

- ▶ Disclosed (message confidentiality)
- ▶ Modified without detection (message integrity)
- ▶ Sent from a masquerading party (message origin authentication)
- ▶ Replayed (uniqueness)

This is usually achieved by encrypting or digitally signing a combination of the message body (or its parts) and header (or its parts).

We need to take a closer look at the two possible forms for protection.

Transport level security

The transport level security has to be applied to the following communication channels:

- ▶ Communication between external service consumers and the secure entry point in the DMZ of ITSOBank, the DataPower appliance.

To secure the communication channel for the service request, HTTPS (using SSL) between the components is implemented. The DataPower appliance terminates the HTTPS session of the service request, as the request will be proxied to WebSphere Application Server residing in the production zone.

- ▶ Communication between the DataPower appliance and WebSphere Application Server.

As the DataPower appliance terminates the HTTPS session of the incoming request, another HTTPS session is used for the communication between the DataPower appliance and WebSphere Application Server. This channel is configured as a mutually authenticated SSL connection. This implies that the

DataPower appliance authenticates the back-end server by validating the server certificate from the WebSphere Application Server and verifying the distinguished name (DN) contained in the certificate. To authenticate the DataPower appliance, the WebSphere Application Server validates the appliance's certificate and verifies its distinguished name.

- ▶ Communication between WebSphere Application Server and the Federated Identity Manager STS.

The WSSM extension and the WS-Trust client used to perform a mapping to the RACF ID are using SOAP for message exchange with the Federated Identity Manager STS. As there are no intermediates between these components, SOAP over HTTPS is considered as the appropriate mechanism to secure these channels. Again, a mutually authenticated connection is implemented.

- ▶ Communication to the IBM Tivoli Directory Server.

The Tivoli Directory Server is the enterprise user registry containing user identities, confidential information like passwords and the RACF ID. To secure the communication between any component and the directory, the services have to be configured to use secure LDAP (LDAPS) to encrypt information from and to the LDAP server.

- ▶ Communication between the Access Manager components.

Access Manager uses SSL for the communication to the Access Manager components by default. There is no additional configuration needed. Only setting up the secure communication to the LDAP server needs additional configuration.

- ▶ Communication between Identity Manager and the adapters for Access Manager for e-business, RACF, and Tivoli Directory Server.

Identity Manager uses specific adapters for the target system to provision user identities. The communication between the adapters and the Identity Manager runtime is secured by using SSL.

- ▶ Communication between WebSphere Application Server and the CICS Transaction Gateway.

The communication between WebSphere Application Server and the CICS Transaction Gateway is secured by using SSL.

Message level protection

A SOAP message travels from the service consumer to the service provider, potentially passing some intermediates along the message path. The intermediate is capable of both receiving and forwarding SOAP messages. When an intermediate receives a SOAP message, it processes the header entries of

the message intended for it and must remove them afterwards before forwarding the message. It may also insert a new header entry for the next intermediate.

Secure protocols, such as SSL/TLS, ensure the security of the message during transmission, but as the messages can be received and forwarded by intermediates, secure end-to-end communication cannot be guaranteed since transport level security has no effect on stored data. Once the message is received and decrypted, *message level security* is needed to protect the message.

As mentioned above, the message path is not controlled by ITSOBank, and the SOAP message has to be encrypted to guarantee that its content is only visible to the service provider and service consumer. To determine from whom the message was sent, to verify the service consumer was who the service consumer claimed to be and to ensure that the data being transmitted was not tampered with, the use of digital certificates and signatures is a requirement from ITSOBank. Message confidentiality and message integrity are implemented by using WS-Security.

Note: For further details on message confidentiality and integrity, please refer to the Web Services Security, SOAP Message Security (WS-Security 2004) at:

<http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

To verify the signatures and to encrypt / decrypt the messages, a security management process needs to be in place to manage and exchange the certificates. Figure 7-15 on page 183 illustrates which certificates are needed at the service consumer and the service provider for signing and encrypting messages to and from the service provider.

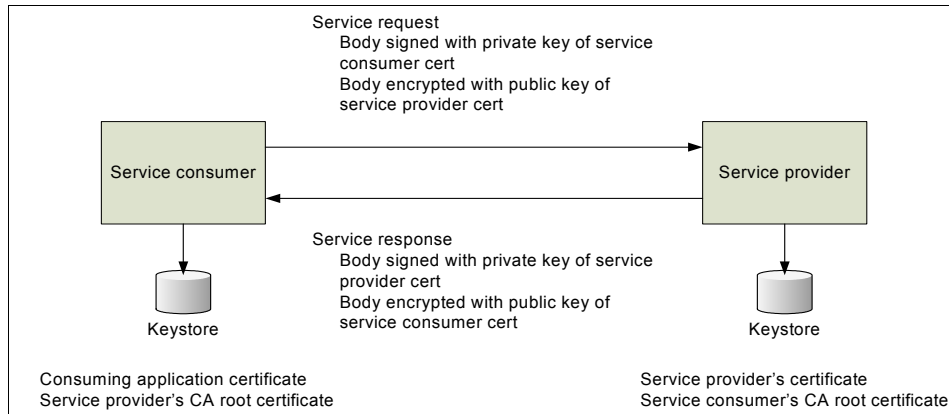


Figure 7-15 Message signature and encryption

As the content of the Web service request message needs to be validated by the DataPower appliance (see security requirement SR9 in 6.3.1, “Security requirements” on page 160), the DataPower appliance decrypts the message and validates the signature. The message will then be signed and encrypted for further processing.

The WebSphere Application Server is required to validate the content of incoming request from the DataPower appliance and internal service consumers. As messages from both the DataPower appliance and internal service consumers are signed and encrypted with certificates from ITSObank, the configuration for both is the same.

The Data and Message Protection services address the following security requirements:

- ▶ SR6: Transport level security
- ▶ SR7: Message integrity
- ▶ SR8: Message confidentiality

7.2.4 Audit Services

To meet the regulatory requirements, auditing should be able to identify the individual user who was performing a transaction at every point in the transaction (see “BR4: Auditing should meet regulatory requirements.” on page 159). The security requirement SR10 requires that every component has to be enabled for auditing and that an infrastructure has to be in place to submit, persistently store, and report on audit data submitted as events.

The Tivoli Common Auditing and Reporting Service is the Tivoli approach to unify IBM product auditing and reporting. Today, it is provided with Access Manager for e-business V6.0, Access Manager for Operating Systems V6.0, and Federated Identity Manager V6.1. The goal is to provide aggregation of events of interest, normalization, and correlation of those events and reporting.

In the Common Auditing and Reporting Service, context auditing is defined as the process of *maintaining detailed, secure logs of critical activities in a business environment*. This includes such items as:

- ▶ Security-related critical activities (login failures, unauthorized access to protected resources, modification of security policy, non-compliance with a specified security policy, and health of security servers)
- ▶ Business-related critical activities (bank transactions)
- ▶ Critical activities related to content management (updates and deletions of critical documents)
- ▶ Change management (changes made by administrators)

Common Auditing and Reporting Service collects the audit data from the enforcement point as well as from other platforms and security applications. To identify which components generate audit events, let us review the current architecture, depicted in Figure 7-16 on page 185, with the relevant security applications and enforcement points.

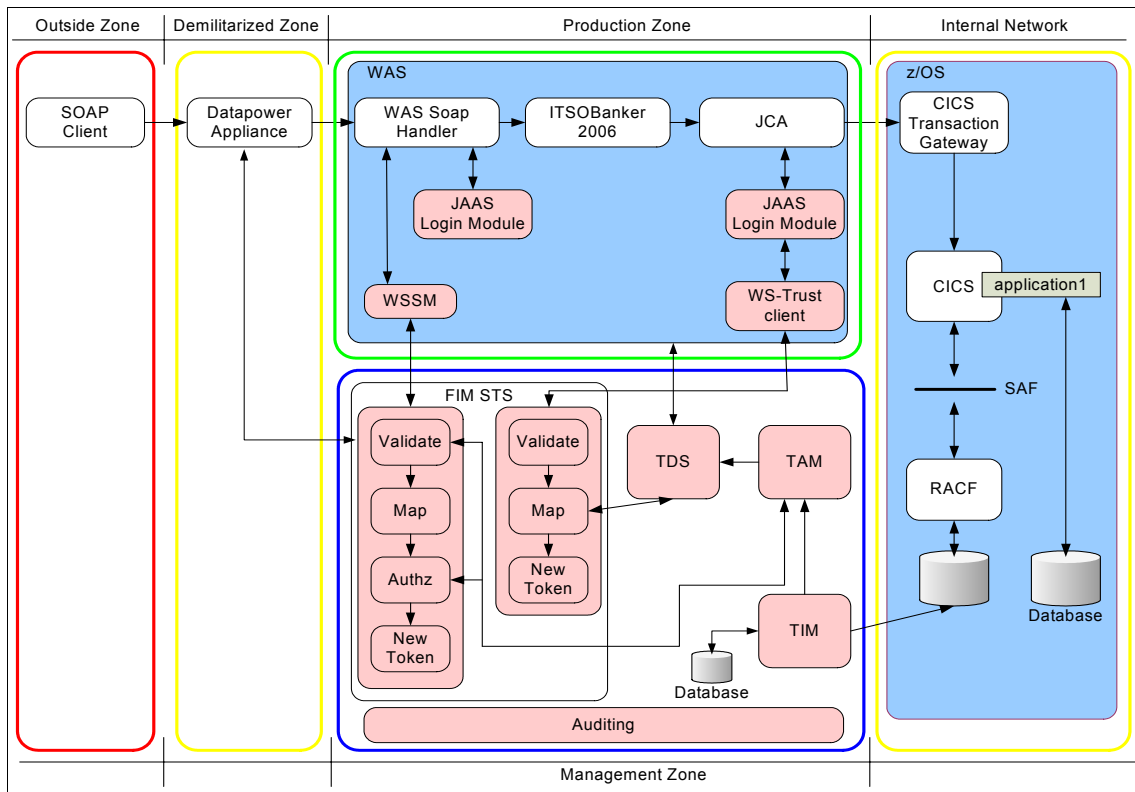


Figure 7-16 Architecture overview

As every authentication and authorization decision has to generate an audit event, the following components are identified to be enabled for auditing:

- ▶ DataPower appliance
- ▶ Federated Identity Manager STS
- ▶ Access Manager for e-business
- ▶ Identity Manager
- ▶ WebSphere Application Server
- ▶ ITSOBank banking application
- ▶ CICS application

The main components for authentication and authorization are Access Manager for e-business and Federated Identity Manager. Both support the common audit report structure for audit logging. As the DataPower appliance is configured to call the Federated Identity Manager STS, the audit events are generated during

the processing of the trust service chain. Authorization for J2EE resources is done by using the JACC provider of Access Manager for e-business, so these events are generated by Access Manager for e-business.

Identity Manager and the CICS application have their own audit services. Identity Manager stores audit events in its relational database and the CICS application keeps its existing audit infrastructure.

7.3 Security Policy Infrastructure

In this section, we cover the different components of the Security Policy Infrastructure and how the ITSOBank scenario reflects them. The Security Policy Infrastructure in the IBM SOA Security Reference Model defines these components:

- ▶ Policy Administration
- ▶ Policy Decision and Enforcement
- ▶ Monitoring and Reporting

7.3.1 Policy Administration

The Policy Administration covers functionality like creating, modifying, import or export of security policies using the available security management tools. The IBM SOA Security Reference Model defines the following categories of security policies. For each category, we map the related security policies derived from the ITSOBank scenario business and security requirements listed in 6.2, “Business requirements” on page 158 and 6.3, “Technical requirements” on page 159.

- ▶ Message protection policies
 - The protocol used to reach the ITSOBank Web service to query the account balance must be HTTPS.
 - A SAML 1.1 security token is required to invoke the Web service from an external service consumer.
 - A user name token is required to invoke the Web service from an internal service consumer.
 - The Web service messages have to be signed and encrypted.
 - The signature method algorithm is RSA-SHA1.
 - The data encryption algorithm is triple DES. To produce the encrypted key, the encryption method RSA is used.

- ▶ Provider policies
 - The user accessing the service has to be authenticated positively.
 - The users accessing the service have to be authorized.
 - Authenticating to RACF must only be possible with the owning user identity.
 - Each component of the solution has to be enabled for auditing and the audit trails have to be able to identify the individual user who was performing a transaction at every point in that transaction.

7.3.2 Policy Decision and Enforcement

The Policy Enforcement Points rely on decisions made by the Policy Decision Points containing the security policies defined in the infrastructure. This section outlines the Policy Decisions Points and Policy Enforcement Points described in the ITSOBank architecture. We step through the individual components listing the roles they play in the context of the individual policies.

- ▶ DataPower appliance

The DataPower appliances enforces message level protection like message integrity and message confidentiality. As it analyzes incoming requests, it is also the Policy Decision Point. For external service requests, it is the PDP and PEP for WS-Security.

- ▶ Federated Identity Manager

Token validation, token exchange, as well as authorizing incoming requests, is done by Federated Identity Manager. Depending on the incoming token type, Federated Identity Manager plays different roles.

- User name token

Validating the user name token includes an authentication of the user identity by Access Manager for e-business. Federated Identity Manager enforces authentication of an incoming Web service request by relying on the decision made by Access Manager for e-business.

- SAML assertions

During the validation process of SAML assertions, Access Manager for e-business is not called. Therefore, it follows that Federated Identity Manager is the decision and enforcement point for authenticating the Web service request.

Federated Identity Manager enforces authorization before the service consumer is granted to invoke the Web service. It relies on the authorization decision made by Access Manager for e-business.

- ▶ WebSphere Application Server
Like the DataPower appliance, the WebSphere Application Server is the policy enforcement and policy decision point to enforce message level security.
- ▶ Access Manager for e-business
Access Manager for e-business is the decision point for authorizing access to requested resources like J2EE resources or Web services. It is also the policy decision point for authenticating incoming Web service requests carrying a user name token.
- ▶ Identity Manager
Identity Manager is the PDP and PEP for provisioning policies.
- ▶ RACF
RACF is the policy decision and policy enforcement point for the CICS application and the CICS Transaction Gateway.

7.3.3 Monitoring and reporting

To check compliance with requirements like availability, performance, and security, the audit events collected by the audit services have to be analyzed. This includes creating reports of significant security events and of the status of the systems and applications.

The security events that have to be included in the reports, among others, are:

- ▶ Failed authentication
- ▶ Failed authorization
- ▶ Which service was invoked by whom
- ▶ Account status, if there is any locked account or password expired

The IBM Tivoli Common Auditing and Reporting Service (CARS) ships with a set of compiled reports, such as:

- ▶ Audit events
- ▶ Authorization events
- ▶ Certificate expiration
- ▶ Administration events
- ▶ Server availability
- ▶ Password changes
- ▶ Resource access

- ▶ Security token service

It is also possible to develop custom reports for specialized requirements.

Using the monitoring reports, compliance with non-functional requirements like performance, availability, and scalability can be verified. Monitoring also includes periodic checks of the log files for particular error conditions.

7.4 Business Security Services

This section describes the solutions needed if ITSOBank is to provide a secure deployment and hosting environment for the ITSOBank business solutions.

7.4.1 Governance, risk, and compliance

Early in the SOA initiative at ITSOBank, a *SOA Governance Board* was formed. From the security perspective, the SOA Governance Board is responsible for defining security standards for interaction with services, establishing roles and responsibilities for policy administration, and how the services of ITSOBank are made available to partners such as ITSOTelco.

The risk management exercise performed by delegates of the SOA Governance Board resulted in the security requirements specified in Chapter 6, “Business scenario” on page 151.

Audit and reporting produces data required for verifying and demonstrating compliance. For example, reports from Tivoli Identity Manager can verify that SR12 (Account recertification) is being met.

7.4.2 Trust Management

The business aspects of Trust Management, such as relationship and liability management, are assumed to be established prior to the development of this technical solution design. Without them, there is no basis to build the cross-enterprise solution from ITSOTelco to ITSOBank.

With the technical aspects of Trust Management, Federated Identity Manager’s key management service is used to manage cryptographic keys used in this solution to provide confidentiality and integrity.

7.4.3 Identity and access

ITSOBank is managing user life cycles by connecting the HR system to the identity management system. The HR system is considered the authoritative resource to feed identities to the repositories and to provide necessary identity information like job roles in specific systems.

The policies listed in 7.3.1, “Policy Administration” on page 186 ensure that an employee owns the right accounts, is a member of the necessary groups, and has access to systems required for his job role. Providing self-care capabilities is also a function of the identity as well as account recertification, which is a requirement from ITSOBank.

7.4.4 Data protection and disclosure control

When using Access Manager for Operating Systems, policies can be created that control who can access specific files on the operating systems, using a defined set of processes. So directories with cryptographic key files, or configuration files, can be accessed only by selected individuals or processes. Also, the audit data of the operating system can be protected with a policy to control who may access it.

7.4.5 Secure systems and networks

This section describes how ITSOBank secures the systems and networks. It is important to know that the concepts mentioned here are best practices and standards and not specific to SOA security.

Note: For further information, we recommend you read Appendix A, “Method for Architecting Secure Solutions”, in *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014.

► Secure networks

The network is designed to address several levels of protection and control and consists of several zones. Between each zone, a firewall is installed.

– Internet (uncontrolled zone)

The Internet is a global network—a network of networks—connecting millions of computers. It cannot be controlled and should not have any components in it.

- Internet DMZ (controlled zone)

The Internet DMZ is generally a controlled zone that contains components with which clients may directly communicate. It provides a *buffer* between the uncontrolled Internet and internal networks. Because this DMZ is typically bounded by two firewalls, there is an opportunity to control traffic at multiple levels:

- Incoming traffic from the Internet to hosts in the DMZ
- Outgoing traffic from hosts in the DMZ to the Internet
- Incoming traffic from internal networks to hosts in the DMZ
- Outgoing traffic from hosts in the DMZ to internal networks

The transport between a controlled and an uncontrolled zone is classified as public. The transport between a controlled and another controlled or a restricted zone is classified as managed.

- Production zone (restricted zone)

One or more network zones may be designated as restricted, that is, they support functions to which access must be strictly controlled, and of course, direct access from an uncontrolled network should not be permitted. As with an Internet DMZ, a restricted network is typically bounded by one or more firewalls and incoming/outgoing traffic may be filtered as appropriate.

The transport between a restricted and a controlled zone is classified as managed. The transport between a restricted and a secured zone is classified as trusted.

- Intranet (controlled zone)

Like the Internet DMZ, the corporate intranet is generally a controlled zone that contains components with which clients may directly communicate. It provides a *buffer* to the internal networks.

- Management zone (secured zone)

One or more network zones may be designated as a secured zone. Access is only available to a small group of authorized staff. Access into one area does not necessarily give you access to another secured area.

The transport into a secured zone is classified as trusted.

- ▶ Secured systems

- ITSOBank uses IBM Tivoli Access Manager for Operating Systems for system hardening like implementing account policies, removing any network service from the system, and controlling access to production servers. Access Manager for Operating Systems also uses the Common Auditing and Reporting Service for auditing and reporting.

► XML Security Gateway

The XML security gateway (DataPower appliance) secures incoming XML and SOAP traffic. It is a first level of defense with these main functions:

- XML/SOAP firewall
- Data validation
- Field level XML security - encrypt and sign individual message fields
- SSL acceleration

7.5 Conclusion

This concludes the basic solution design for the planned SOA project at ITSOBank. All requirements have been documented; the business requirements are based on a risk management and mitigation approach, and the functional and non-functional requirements are based on applying the IBM SOA Security Reference Model.

In Chapter 8, “Technical implementation” on page 193, we walk through a step-by-step process of implementing some of the technical details.



Technical implementation

This chapter contains configuration instructions for implementing the solution architecture described in the preceding chapter. It is assumed that the reader is familiar with the products being used. Installation steps and basic product configuration are not described.

This chapter covers the following aspects of the technical implementation:

- ▶ Implementation scope
- ▶ Configure security for the ITSO Banking Application
- ▶ Deploying the ITSO Banking Application
- ▶ Configure Web Service Security Management
- ▶ Running the scenario
- ▶ Common Auditing and Reporting Service configuration

8.1 Implementation scope

This section describes the scope of the implementation of the ITSOBank example scenario. The implementation reflects parts of Chapter 7, “Solution design” on page 163. It shows how an internal client invokes the Web service.

Figure 8-1 shows the sequence of steps that occur when invoking the Web service.

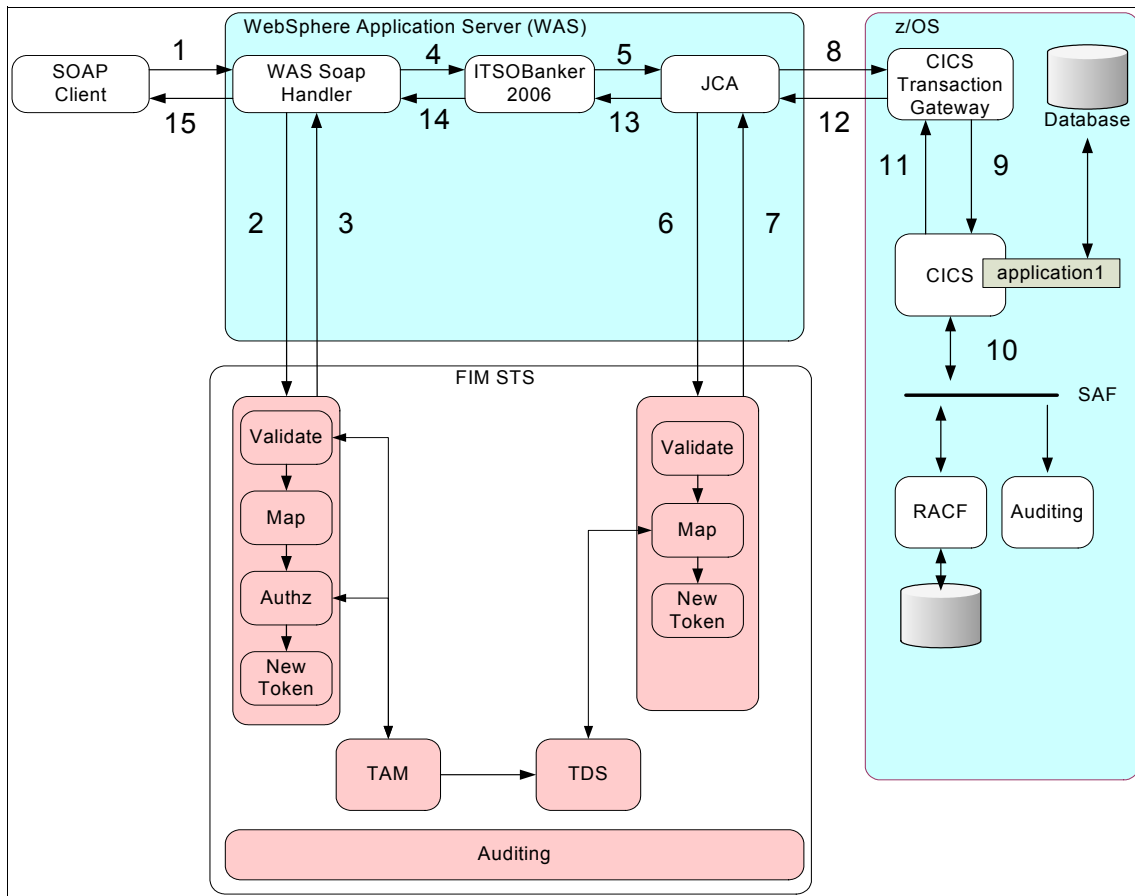


Figure 8-1 Scope of the implementation

The flow when a user accesses the Web service is summarized here.

1. The SOAP client invokes the Web service by sending a request to the WebSphere Application Server. The message carries a *user name token* and the message is *signed* and *encrypted*.

2. The WebSphere Application Server SOAP handler processes the message and invokes Federated Identity Manager WSSM for authentication and authorization. This sends a *SecurityTokenRequest* to the Federated Identity Manager trust service.

The Federated Identity Manager trust service now performs the following tasks:

- a. It validates the incoming user name token and authenticates the incoming user ID by delegating the authentication to Access Manager.
 - b. It maps the incoming user ID to the internal user ID.
 - c. It authorizes the Web service call. This step is also delegated to Access Manager.
 - d. It issues a SAML assertion based on the mapped identity.
3. The SAML token issued by Federated Identity Manager is returned as a *SecurityTokenResponse*.
 4. After the user is successfully authenticated, the application is called.
 5. The application uses the CICS JCA to communication with the CICS Transaction Gateway. The JCA adapter invokes a JAAS login module to obtain user credentials to send to the CICS Transaction Gateway.
 6. The JAAS login module calls the Federated Identity Manager trust service by sending a *SecurityTokenRequest*.

The Federated Identity Manager trust service now performs the following tasks:

- a. It validates the incoming user name token.
 - b. It queries the LDAP to retrieve the RACF ID of the current user.
 - c. It issues a Passticket and generates a user name token to return.
7. The Federated Identity Manager trust service returns the user name token in a *SecurityTokenResponse*.
 8. The CICS Transaction Gateway is called.
 9. The CICS Transaction Gateway forwards the request to the CICS application.
 10. Authentication, authorization, and auditing with the real user ID is performed by RACF.
 11. The current balance is returned to CICS Transaction Gateway.
 12. The current balance is returned to the JCA connector.
 13. The current balance is returned to the EJB.

14. The current balance is returned to the WebSphere Application Server SOAP Handler, which creates the Web service response, including applying the appropriate message protection.
15. The response is sent to the SOAP client.

8.2 Configure security for the ITSO Banking Application

The steps in this section might typically be executed by a security administrator, who is responsible for adding *declarative security constraints* to applications prior to their deployment. In this case, the Rational® Software Architect (RSA) is used, though it is also possible to use Rational Application Developer (RAD).

The assumption is that the application development team has constructed the application, and made available to the security administrator other EAR files that represent the J2EE application and its application client. The security administrator will complete the application construction by adding declarative security constraints that meet the business and technical requirements. The resulting versions of the applications are exported from the Rational Software Architect environment and provided to operational personnel who will deploy the applications in an application server environment.

Note: More information about securing Web services can be found in the redbook *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257.

8.2.1 Import the application into Rational Software Architect

Start the IBM Rational Software Architect environment, and view the current workspace using the J2EE perspective, as shown in Figure 8-2 on page 197.

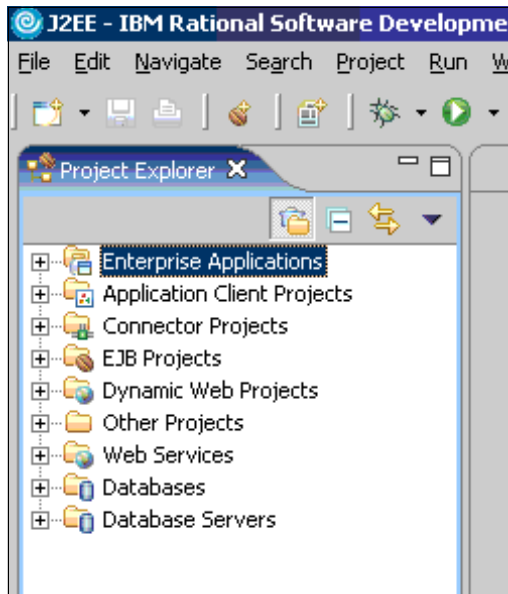


Figure 8-2 IBM Rational Software Architect using the J2EE perspective

Right-click the **Enterprise Applications** container, and select **Import...** → **EAR file**.

Note: Instructions on how to obtain the unsecured versions of the sample application are available in Appendix D, "Additional material" on page 389.

For the application client (ITSOBankerClient2006.ear), locate the EAR file on the file system and accept the defaults in the import dialogs to complete the import process. The client application contains an application module, named ITSOBankerAppClient. The resulting project view in Rational Software Architect is depicted in Figure 8-3.

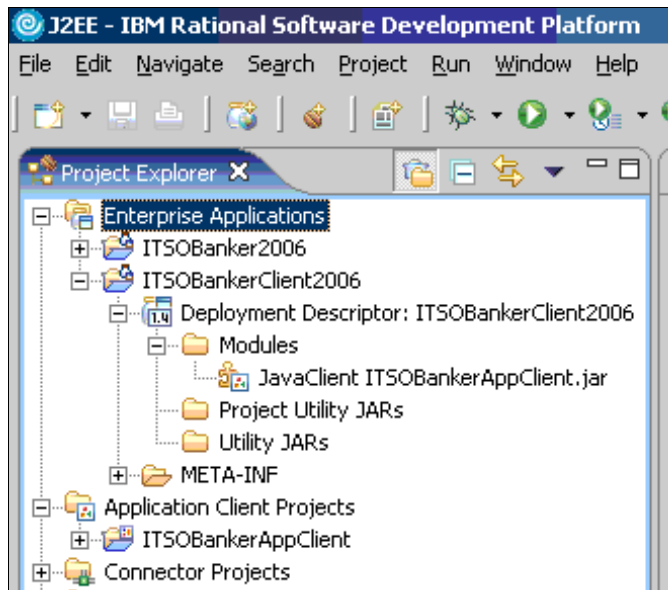


Figure 8-3 J2EE perspective after successfully importing ITSO Banker Application client

Repeat the import steps for the application containing the Web service (ITSOBanker2006.ear).

It is possible that a build error is seen with the AccountEJB project, where the CICS resource adapter classes cannot be resolved. To correct this, right-click the **AccountEJB** project and choose **Properties**. Select the **Java Build Path** property set, and check the **cicseciConnector** in the Projects tab, as shown in Figure 8-4 on page 199.

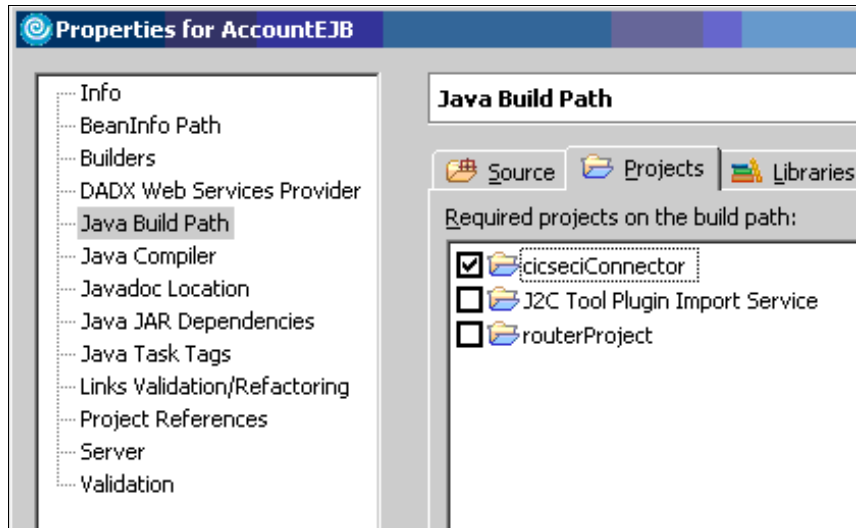


Figure 8-4 Correcting the Java build path for the AccountEJB project

If automatic building of projects is enabled in the Rational Software Architect environment, recompilation will begin immediately; otherwise, the **Project** → **Build All** option can be used to start the recompilation.

The build problems should disappear and the Project Explorer should show the ITSOBanker2006 application with no errors, as shown in Figure 8-5.

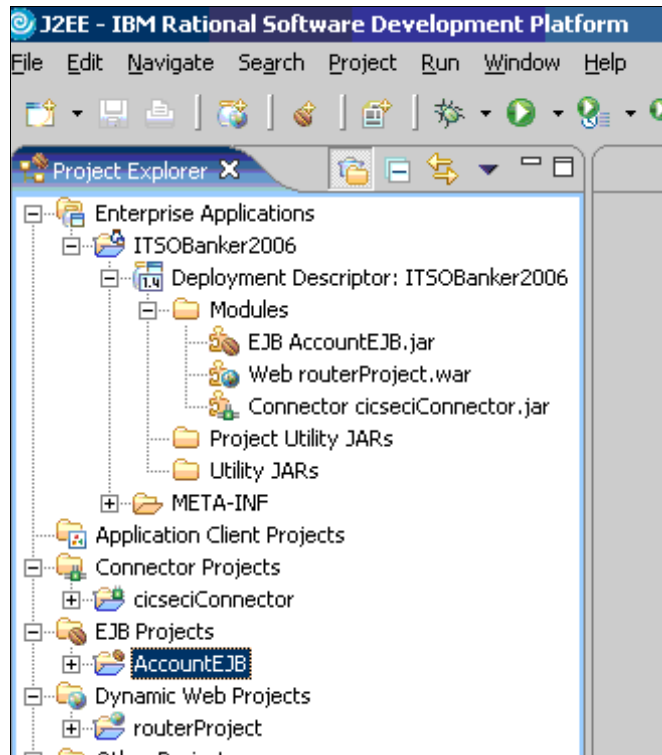


Figure 8-5 ITSOBanker2006 application successfully imported

As a final verification step, expand the **Web Services** container in the Project Explorer.

Web services implemented in active J2EE projects are displayed in the Services folder, depicted in Figure 8-6 on page 201. The AccountJCA service should be present, implemented by the *AccountJCA* service class. This is the Web service implemented and exposed in the ITSOBanker2006 application.

Application modules that invoke Web services are contained in the Clients folder. The ITSOBankerAppClient should be visible, indicating that this J2EE module invokes the Web service named *service/AccountJCAService*.

Note that the same application may appear in both the Services and Clients folders if it implements Web services and also invokes other Web services.

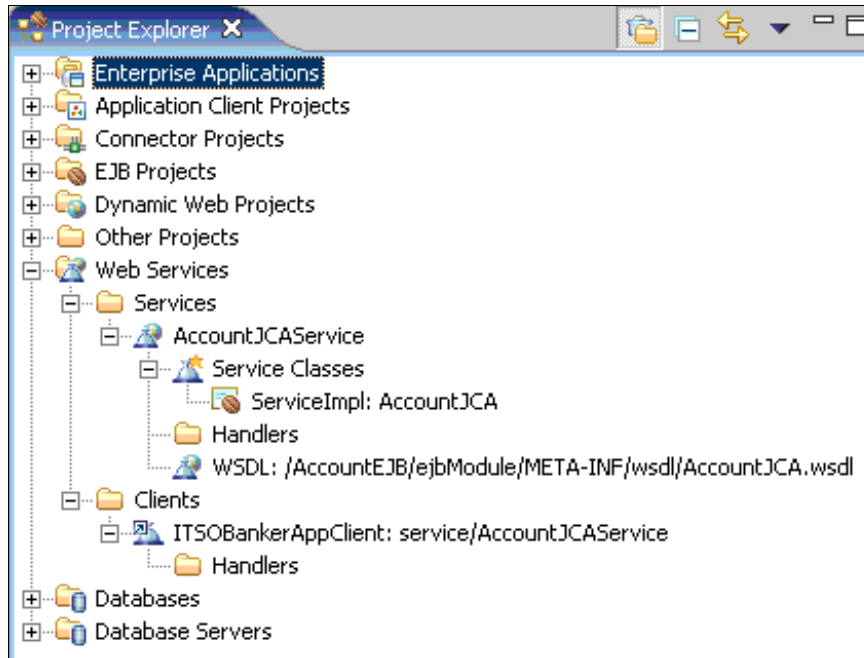


Figure 8-6 Web services implemented and referenced in the Rational Software Architect environment

8.2.2 Key stores

X.509 certificates are required for the cryptographic operations used to protect the Web services messages. The application client and the application have their own *key stores*. All key stores used in this example are in JKS format. The key store password used in all cases is *itso*.

Client key store

The client key store is named `client.jks` and contains the following keys:

- ▶ Client's key pair
- ▶ Application's public key

Application key store

The application key store is named `was.jks` and contains the following keys:

- ▶ Application's key pair
- ▶ Client's public key

Note: Instructions on how to obtain sample key stores are available in Appendix D, “Additional material” on page 389.

8.2.3 Configure the application client

Open the deployment descriptor for the ITSOBankAppClient project by double-clicking it in the Project Explorer view, as shown in Figure 8-7.

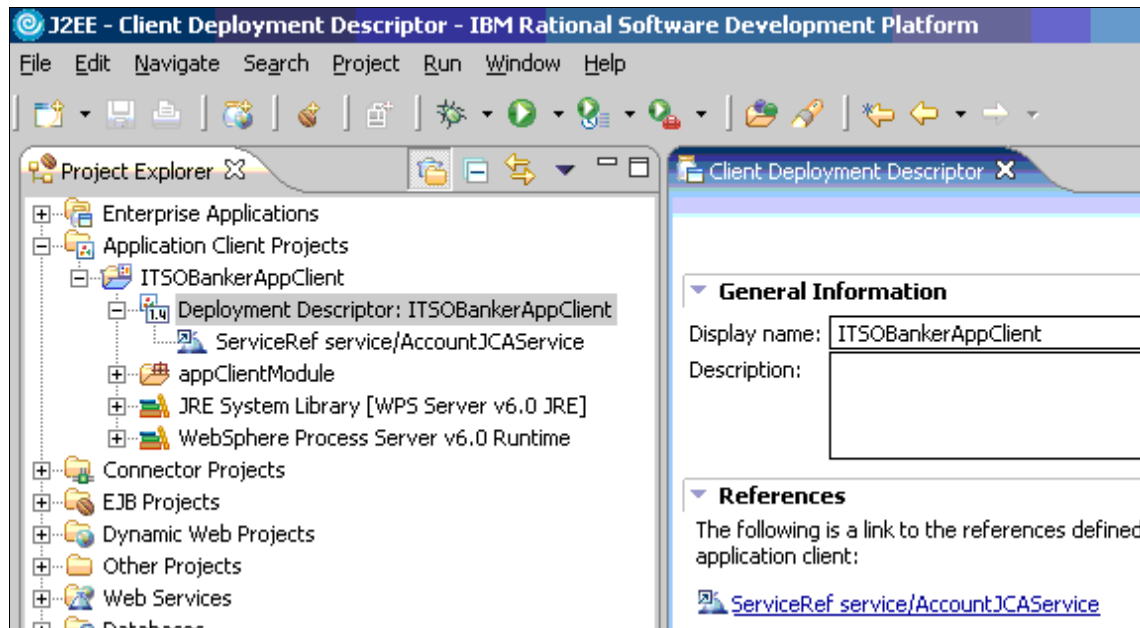


Figure 8-7 Application client deployment descriptor

Request generation

In this section, the following security constraints are added to the application client:

- ▶ Sign Web service request with the client's private key.
- ▶ Encrypt Web service request with the application's public key.
- ▶ Solicit user name and password from the user.
- ▶ Include a user name token in the Web service request.

Navigate to the *WS Extension* tab of the deployment descriptor, shown in Figure 8-8 on page 203. This is where the security requirements are described in more abstract terms.

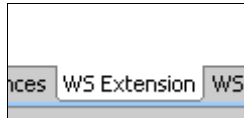


Figure 8-8 WS Extension tab of an application client deployment descriptor

The first stage of the procedure is to configure signing of the outgoing message and security token.

Select the **AccountJCA** entry in the Port QName Bindings section. Then, expand the **Request Generator Configuration** section, and expand the **Security Token** section within, as depicted in Figure 8-9.

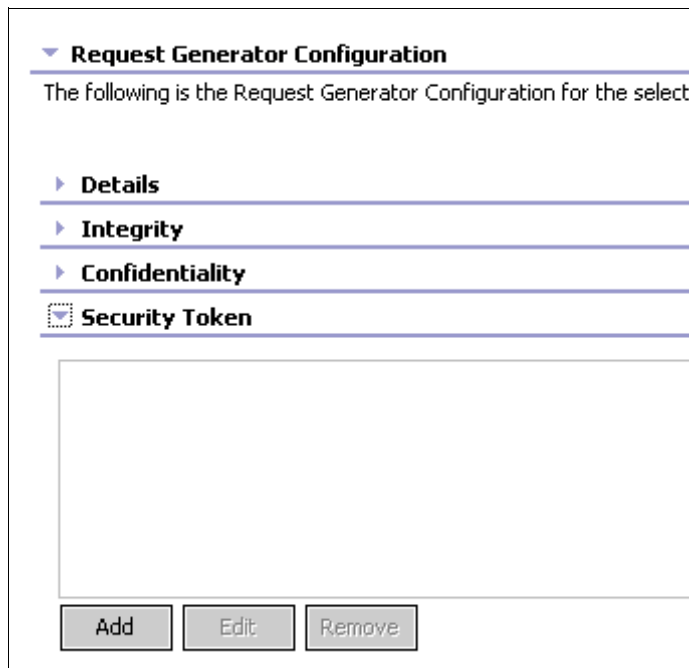


Figure 8-9 Request Generator → Security Token configuration section

Click **Add** to add a new security token definition. In this case, shown in Figure 8-10, we add a user name token by filling in the Name field and selecting a **Username** token type. Note that the Local name field is filled in based on the selected token type.

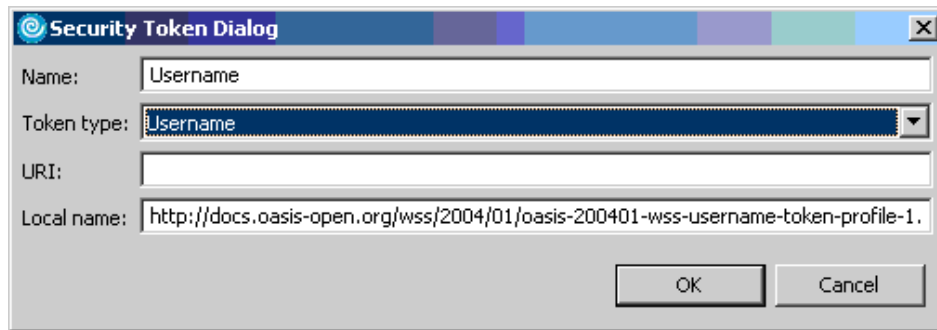


Figure 8-10 Creating a user name token security token requirement

Navigate to the WS Binding tab. This is where the configurations on the WS Extension tab are bound to concrete objects, like key store, certificates, and classes that implement token generation and consumption. The configuration in this major section spans both extensions and bindings.

Expand the **Security Request Generator Binding Configuration** section, and the **Token Generator** section within, as shown in Figure 8-11 on page 205. Click **Add**.

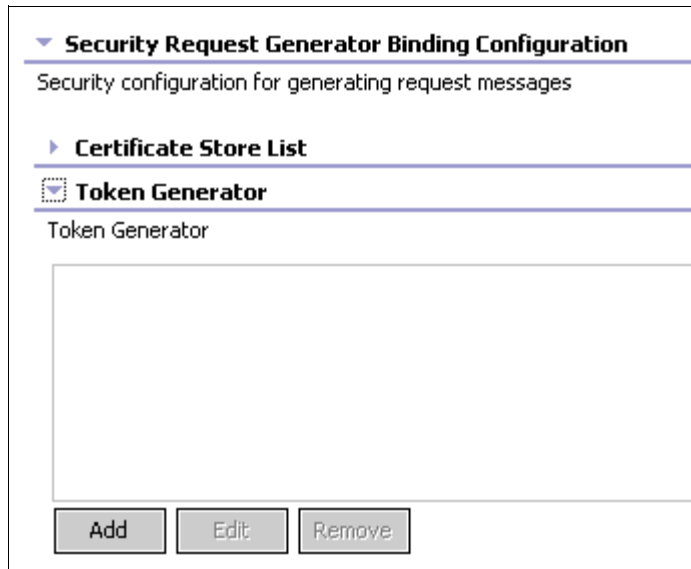


Figure 8-11 Security Request Generator Binding → Token Generator configuration section

Fill in the values according to Table 8-1 into the dialog shown in Figure 8-12 on page 206.

Table 8-1 Token Generator values

Parameter	Value
Token generator name	Username
Token generator class	com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator
Security token	Username
Use value type	Checked
Value type	Username Token
Call back handler	com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler

Note that the *call back handler* chosen will display a dialog box to the user to solicit their user name and password. This call back handler is appropriate because the client application is an AWT application.

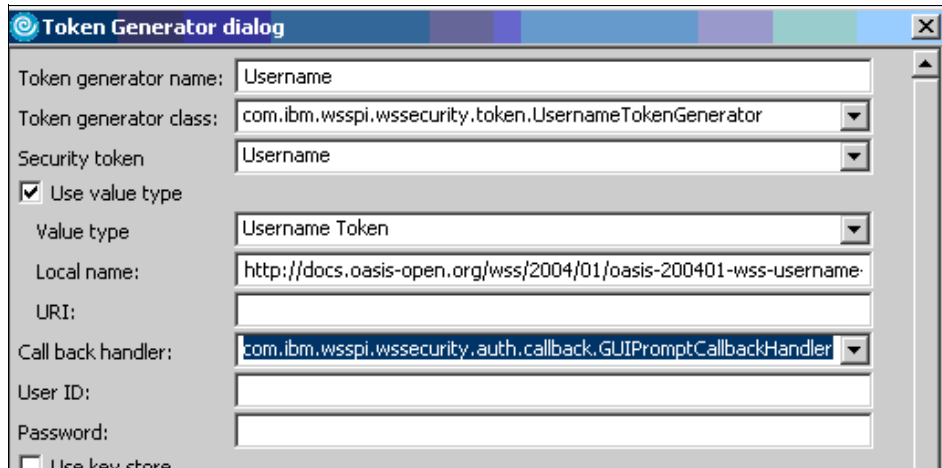


Figure 8-12 User name token generator

Return to the WS Extension tab and expand the **Integrity** configuration section.

Add a new Integrity configuration to specify signing of the message and the security token, as shown in Figure 8-13. For both entries, the *message parts dialect* value is:

<http://www.ibm.com/websphere/webservices/wssecurity/dialect-was>

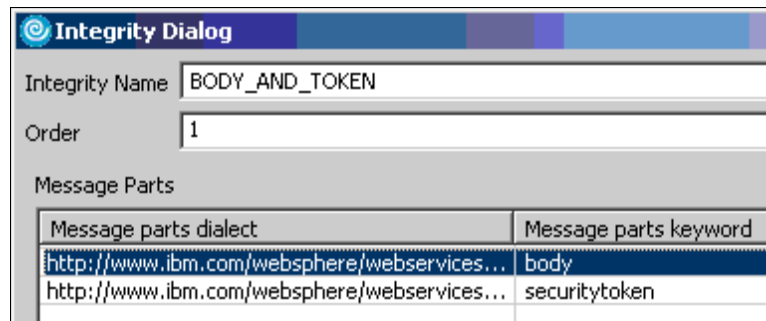


Figure 8-13 Specifying an integrity configuration

Return to the WS-Binding tab to configure a token generator for the token that will sign the message. Select **Token Generator** → **Username** → **Add**. This is an X.509 token generator, and the private key to use to sign the message is specified in this dialog box. Notice use of the key store introduced in 8.2.2, “Key stores” on page 201.

Note that you can populate the Local name parameter automatically by first choosing **X.509 certificate** token from the Value type drop-down menu, before

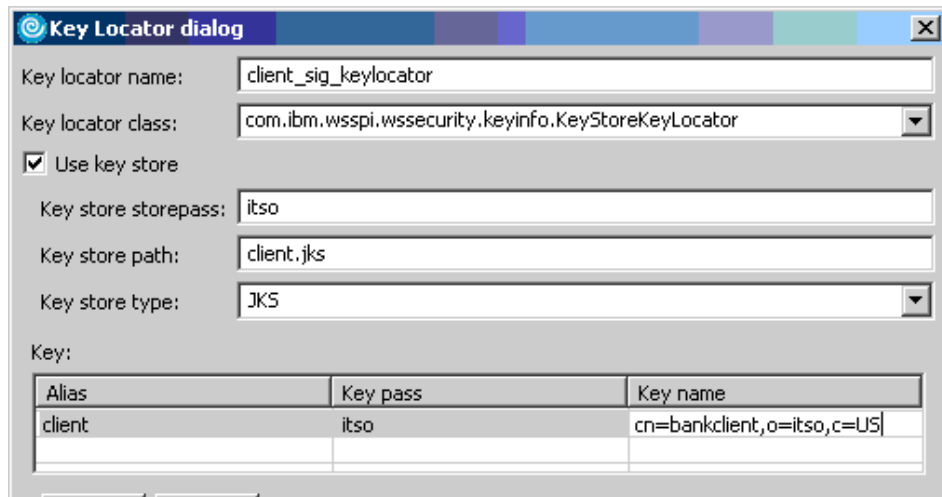
overwriting its value with SigningToken_vtype. This dialog is depicted in Figure 8-14.

Attention: Note that the key stores do not need to be available at this time. No validation is performed by Rational Software Architect that the key store exists or the key store password is valid.

Alias	Key pass	Key name
client	itso	cn=bankclient,o=itso,c=US

Figure 8-14 Token generator for X.509 token to sign the message

Next, create a Key Locator for the signing token according to the dialog in Figure 8-15.



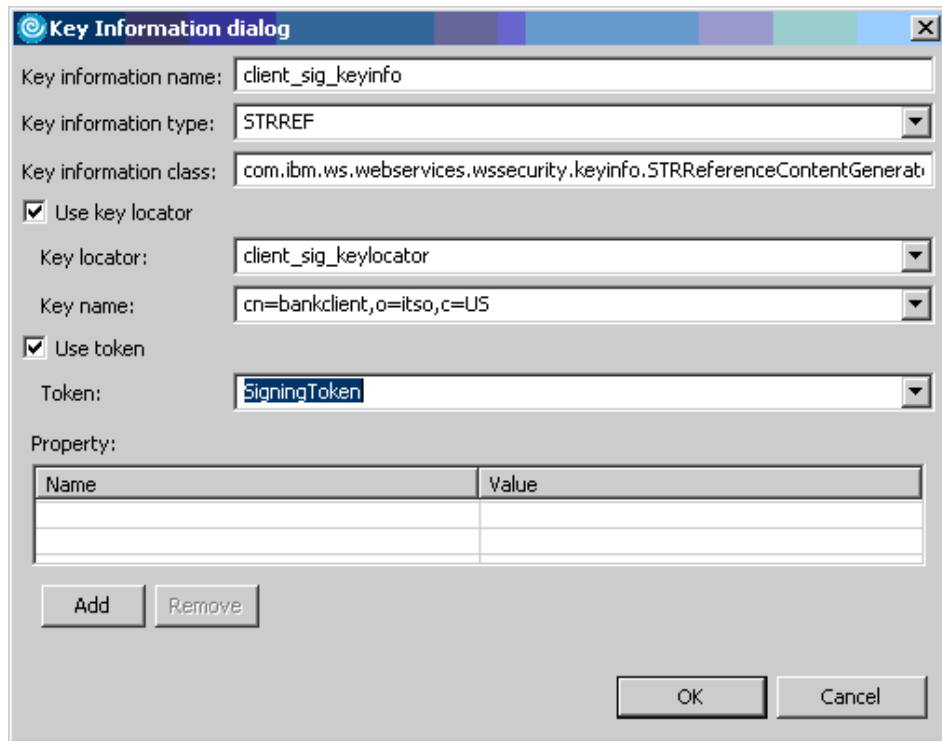
The image shows a 'Key Locator dialog' window with the following fields and controls:

- Key locator name:
- Key locator class:
- Use key store
- Key store storepass:
- Key store path:
- Key store type:
- Key:

Alias	Key pass	Key name
client	itso	cn=bankclient,o=itso,c=US

Figure 8-15 Key locator for the signing key

The final step in identifying a key is the Key Information. This provides a level of indirection between the keys/certificates and the consumers of those artifacts. It also links the key locator with the token generator. Provide the information as shown in Figure 8-16.



The image shows a 'Key Information dialog' window with the following fields and options:

- Key information name: client_sig_keyinfo
- Key information type: STRREF
- Key information class: com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentGenerat
- Use key locator
 - Key locator: client_sig_keylocator
 - Key name: cn=bankclient,o=itso,c=US
- Use token
 - Token: SigningToken
- Property:

Name	Value

Buttons: Add, Remove, OK, Cancel

Figure 8-16 Key information for the signing key

Expand the **Signing Information** configuration section, and add a new Signing Information configuration. Notice that this entry references the key locator and key information just created. This dialog is depicted in Figure 8-17.

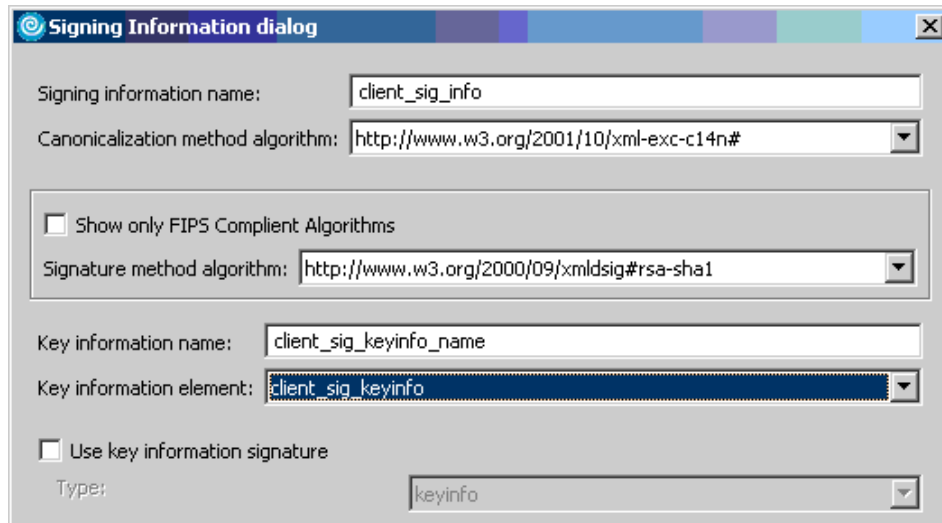


Figure 8-17 Signing information for client's Web service request

With the newly created signing information selected, add a new Part Reference entry, as shown in Figure 8-18. Notice that it binds the *BODY_AND_TOKEN* integrity configuration described on the WS Extensions tab to the signing information being created on the WS Bindings tab.

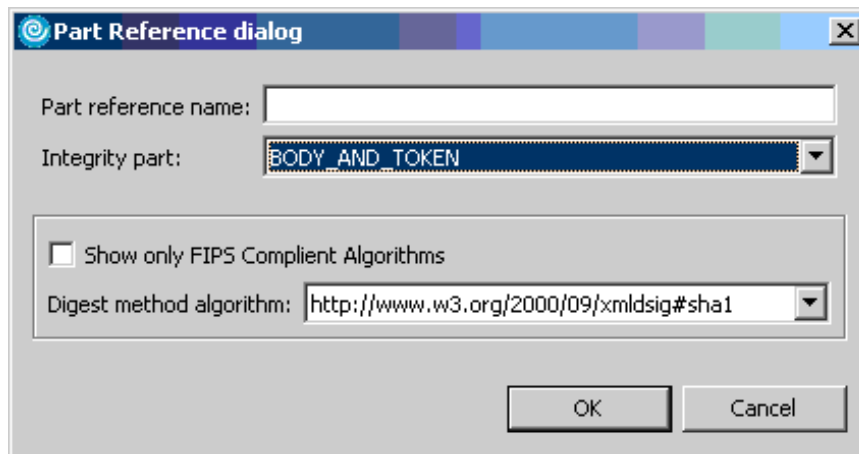


Figure 8-18 Specifying which parts of the message will be signed

The last step in the message signing configuration is to add a Transform entry. With the newly created part reference selected, add a new Transform, as shown in Figure 8-19.

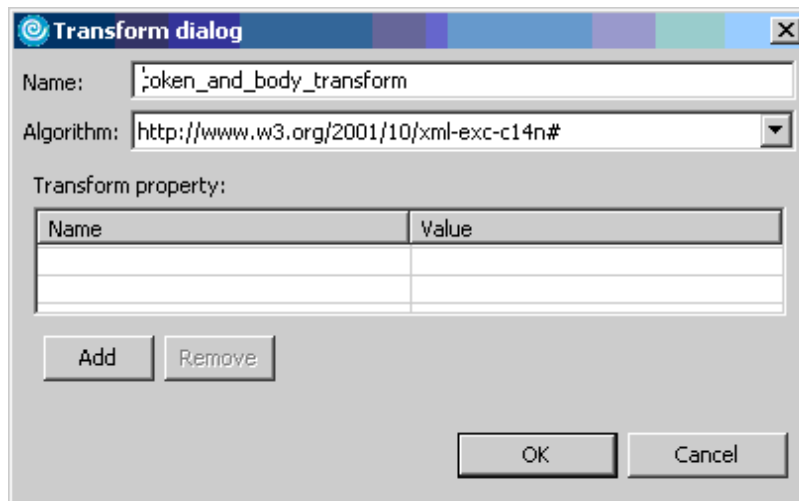


Figure 8-19 Transform configuration for the part reference

With the configuration for signing of the requests completed, the next steps in the procedure configure encryption of the requests.

Return to the WS Extension tab, and expand the **Confidentiality** section. Select **Port QName Bindings** → **Account JCA**. Add a new confidentiality entry, indicating that the content of the message body and the user name token should both be encrypted, as shown in Figure 8-20. For both entries, the message parts dialect value is:

<http://www.ibm.com/websphere/webservices/wssecurity/ dialect-was>

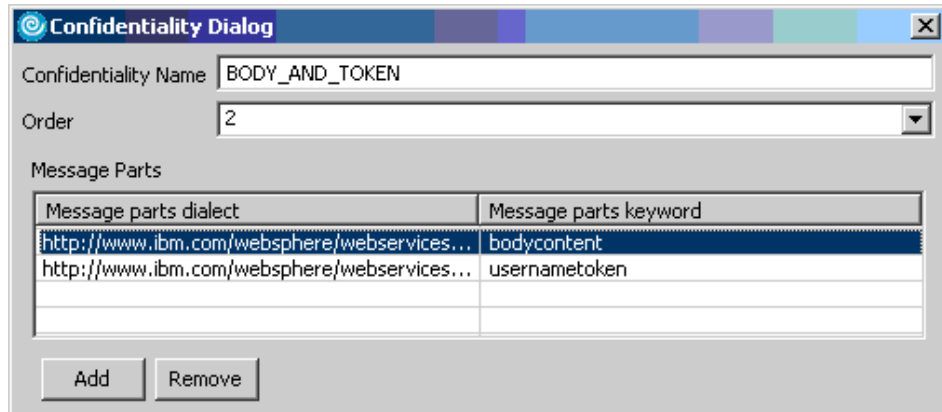


Figure 8-20 Adding confidentiality configuration for the requests

On the WS Bindings tab, add a key locator for the key used to encrypt the message, depicted in Figure 8-21. This is the public key of the Web service in this case.

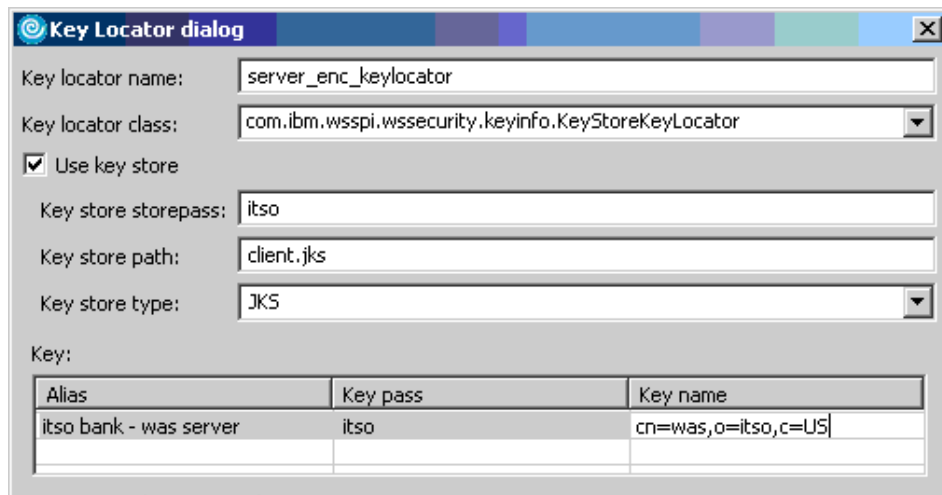


Figure 8-21 Key locator for request encryption key

Add key information for the encryption key (Figure 8-22). The key information type is KEYID instead of STRREF (which was used for the signing key).

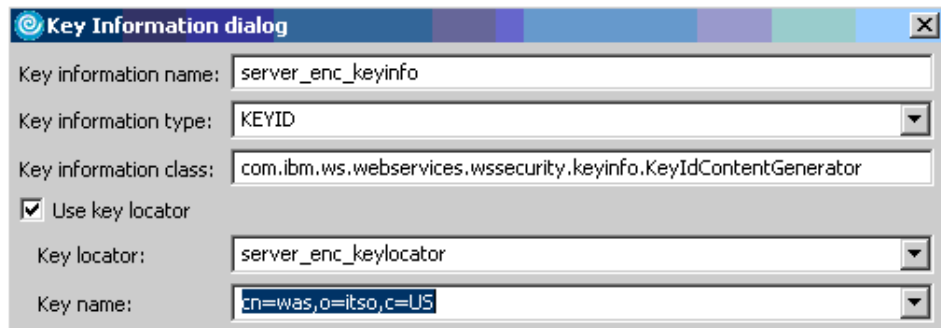


Figure 8-22 Key information for encryption key

Lastly, add Encryption Information to link the desire for request confidentiality with the encryption key to be used, as shown in Figure 8-23.

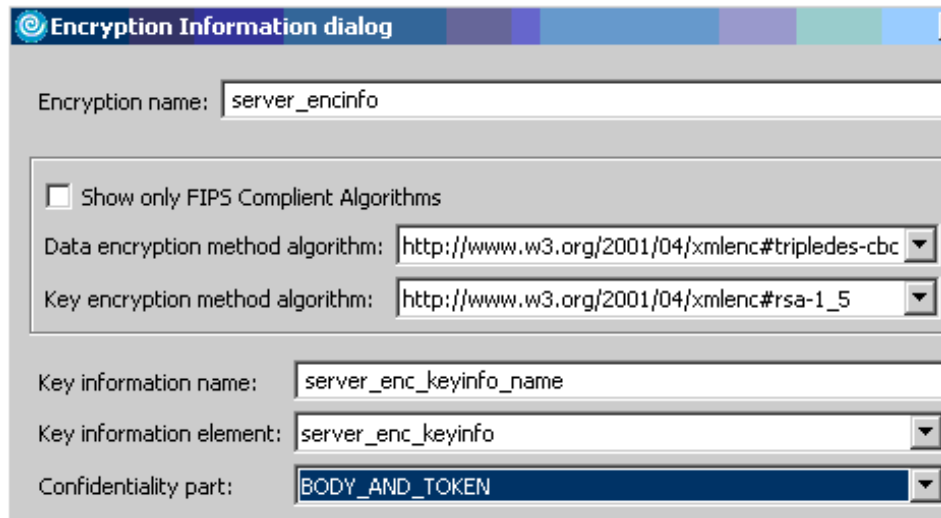


Figure 8-23 Encryption information for requests

The configuration to inject the required security information into outgoing Web services requests is now complete.

Save the deployment descriptor now.

Response consumption

The application client also requires configuration to be able to process responses expected from the Web service. This configuration includes:

- ▶ Decrypt Web service response using client's private key.
- ▶ Validate signature of Web service response using application's public key.

Begin by navigating to the **WS Extension** tab. Expand the **Security Response Consumer Binding Configuration** section.

Expand the **Required Integrity** section. Add a constraint to require an integrity checksum for the message body, as shown in Figure 8-24. For the message part entry, the message parts dialect value should be:

<http://www.ibm.com/websphere/webservices/wssecurity/dialect-was>

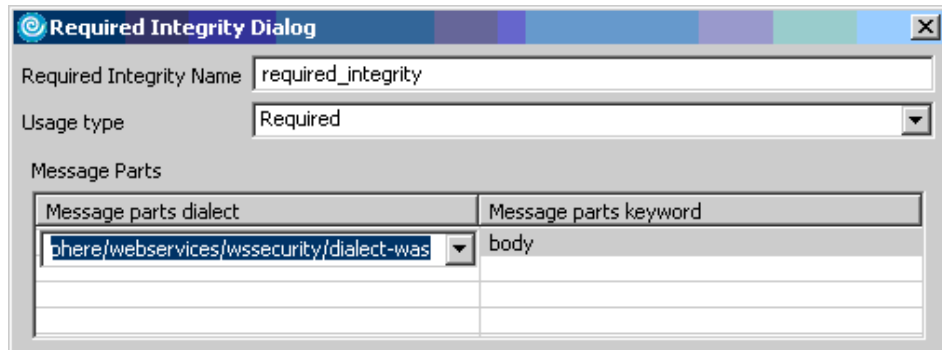


Figure 8-24 Require integrity for the body of responses

Expand the **Required Confidentiality** section. Add a confidentiality requirement as well for the content of the body of the response, depicted in Figure 8-25 on page 215. For the message part entry, the message parts dialect value should be:

<http://www.ibm.com/websphere/webservices/wssecurity/dialect-was>

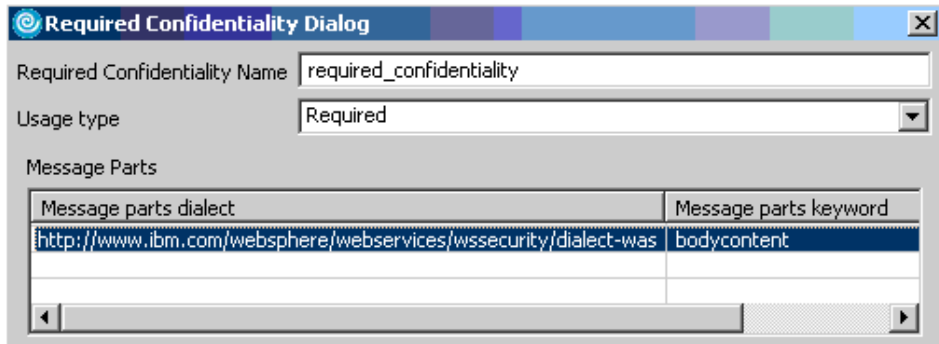


Figure 8-25 Add requirement for confidentiality of the bodycontent

Next, navigate to the *WS Binding* tab of the deployment descriptor. This is where the requirements for integrity and confidentiality are bound to keys that can be used for signature validation and decryption.

Expand the **Security Response Consumer Binding Configuration** section, and the **Trust Anchor** section within.

Add a new trust anchor definition to specify the file name and password for the key store used by the client, as shown in Figure 8-26.

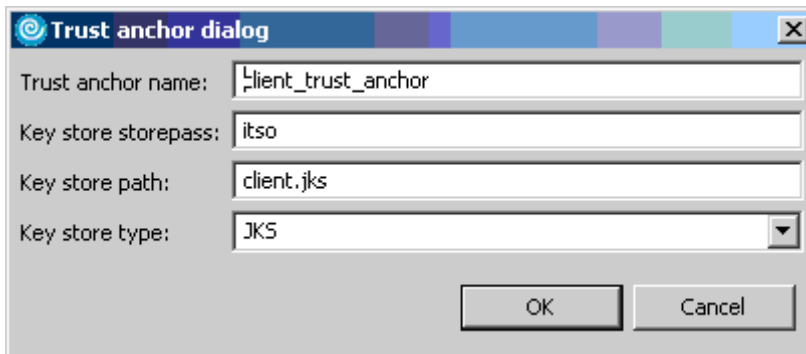
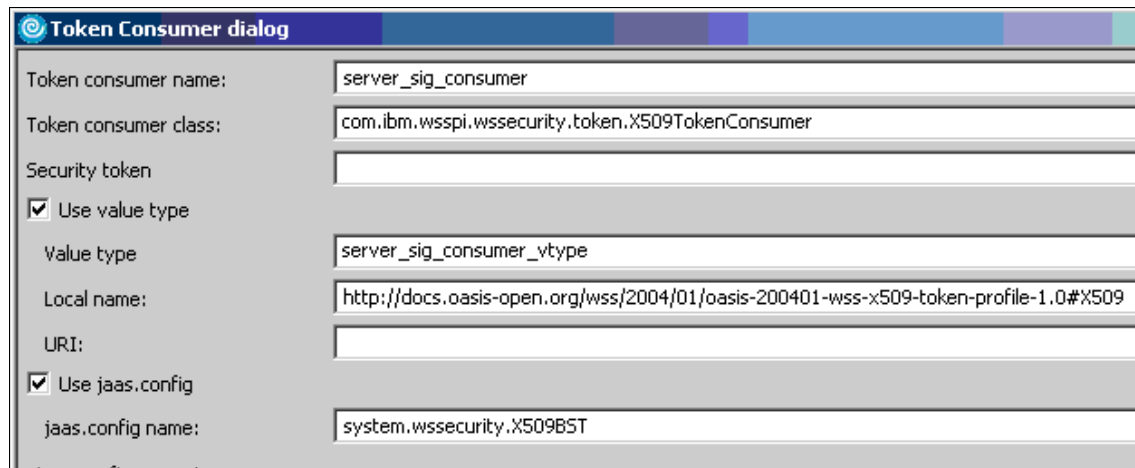


Figure 8-26 Client trust anchor

Expand the **Token Consumer** configuration section, and add a new token consumer for the X.509 certificate used for validation of the signature on the responses, as depicted in Figure 8-27. The JAAS login value chosen, namely *system.wssecurity.X509BST*, is a standard token consumer in WebSphere Application Server for processing X.509 binary security tokens.

Note: The value for the Local name attribute can be automatically populated by choosing the **X509 certificate token** value type. The full value for the Local name is:

`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509`



Token consumer name:	server_sig_consumer
Token consumer class:	com.ibm.wsspi.wssecurity.token.X509TokenConsumer
Security token	
<input checked="" type="checkbox"/> Use value type	
Value type	server_sig_consumer_vtype
Local name:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
URI:	
<input checked="" type="checkbox"/> Use jaas.config	
jaas.config name:	system.wssecurity.X509BST

Figure 8-27 Token consumer for server signature validation

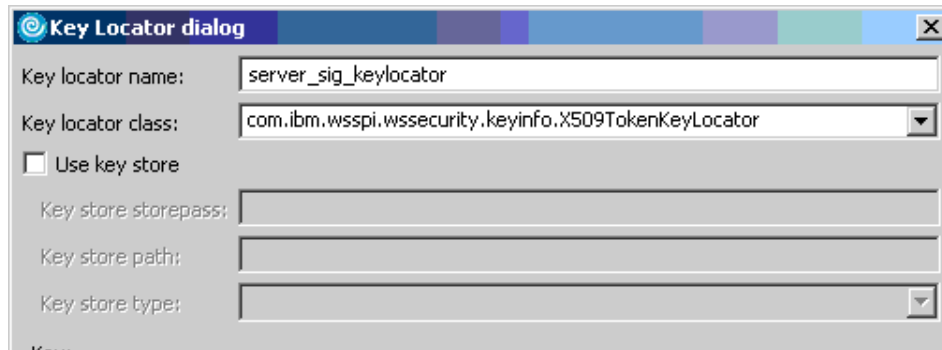
Also, create a token consumer configuration for the key used to decrypt the response (Figure 8-28 on page 217).

The image shows a software dialog box titled "Token Consumer dialog". It contains several configuration fields:

- Token consumer name: client_enc_consumer
- Token consumer class: com.ibm.wsspi.wssecurity.token.X509TokenConsumer
- Security token: (empty field)
- Use value type
- Value type: client_enc_consumer_vtype
- Local name: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
- URI: (empty field)
- Use jaas.config
- jaas.config name: system.wssecurity.X509BST

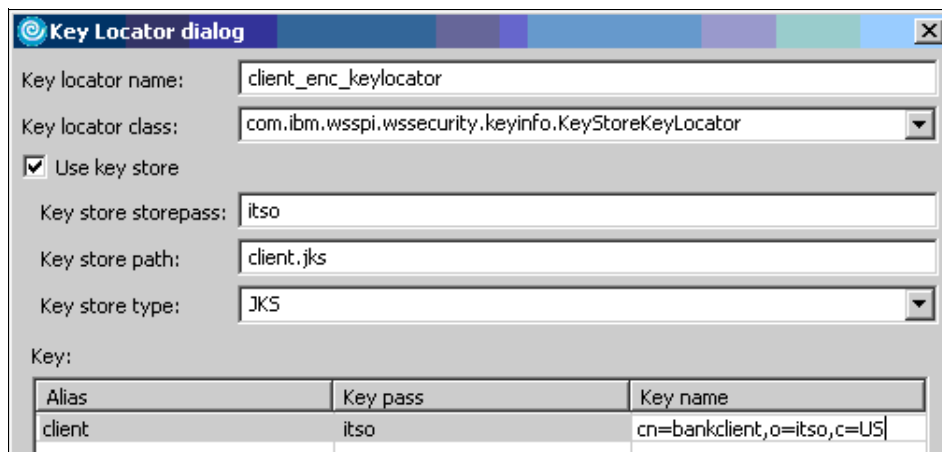
Figure 8-28 Token consumer for message decryption

Key locators are required to specify the location of keys used in the signature validation and decryption processes. Create these key locators according to the screen shots in Figure 8-29 and Figure 8-30.



The screenshot shows a 'Key Locator dialog' window. The 'Key locator name' field contains 'server_sig_keylocator'. The 'Key locator class' dropdown menu is set to 'com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator'. The 'Use key store' checkbox is unchecked. The 'Key store storepass', 'Key store path', and 'Key store type' fields are empty.

Figure 8-29 Key locator for signature validation

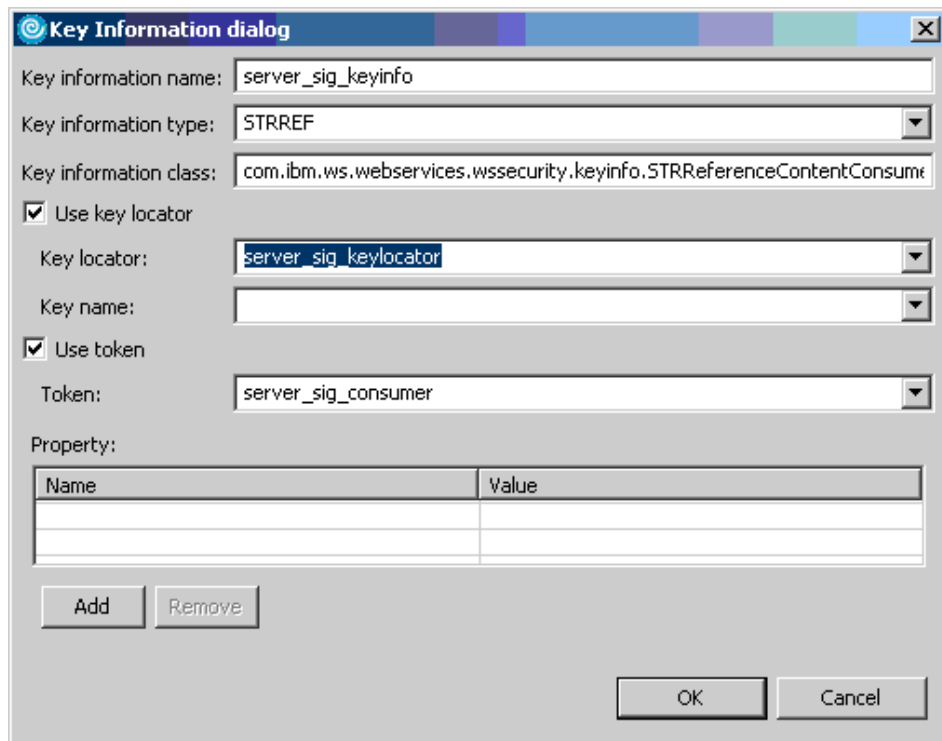


The screenshot shows a 'Key Locator dialog' window. The 'Key locator name' field contains 'client_enc_keylocator'. The 'Key locator class' dropdown menu is set to 'com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator'. The 'Use key store' checkbox is checked. The 'Key store storepass' field contains 'itso'. The 'Key store path' field contains 'client.jks'. The 'Key store type' dropdown menu is set to 'JKS'. Below these fields is a 'Key:' section with a table:

Alias	Key pass	Key name
client	itso	cn=bankclient,o=itso,c=US

Figure 8-30 Key locator for decryption

Expand the **Key Information** configuration section. Add key information configuration for signature validation, as in Figure 8-31, and decryption, as in Figure 8-32 on page 220.



The image shows a 'Key Information dialog' window with the following fields and options:

- Key information name:
- Key information type:
- Key information class:
- Use key locator
- Key locator:
- Key name:
- Use token
- Token:
- Property:

Name	Value
- Buttons: Add, Remove, OK, Cancel

Figure 8-31 Key information for signature validation

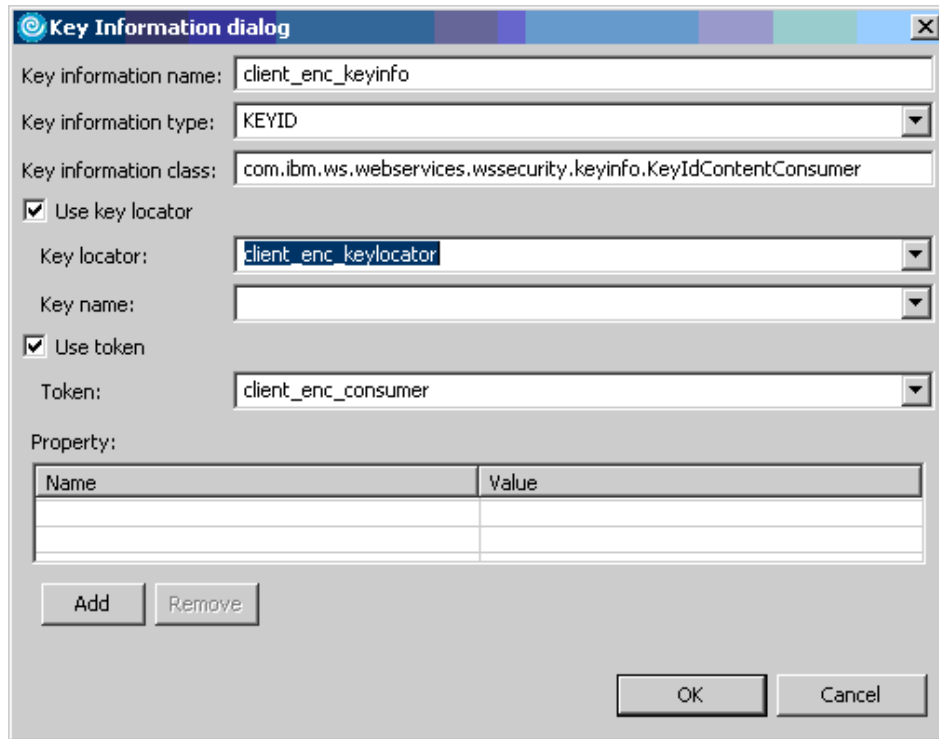


Figure 8-32 Key information for decryption

Expand the **Signing Information** section, and add an entry to specify the key information and location that will be used to validate the signature on the Web service responses (Figure 8-33 on page 221).

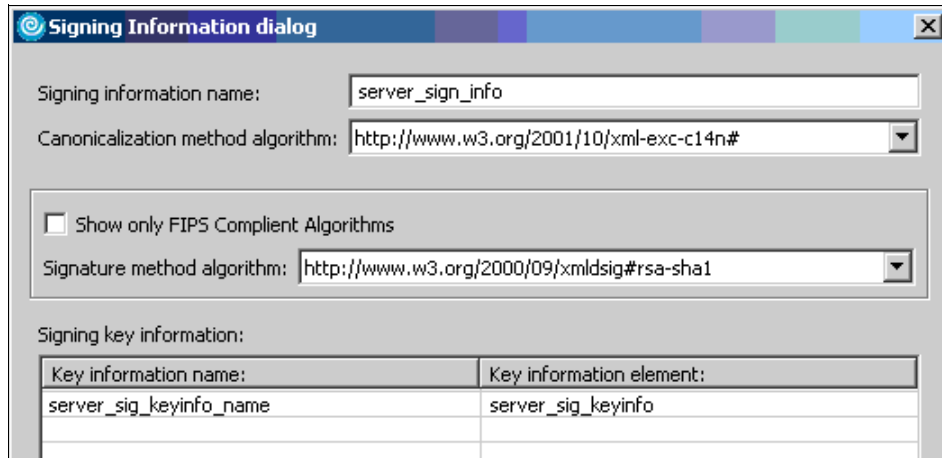


Figure 8-33 Binding configuration for signing information

Expand the **Part References** section. Select the newly created signing information, and click to add a new part reference, shown in Figure 8-34. This links the signing information with the integrity constraint defined earlier on the *WS Extensions* tab.

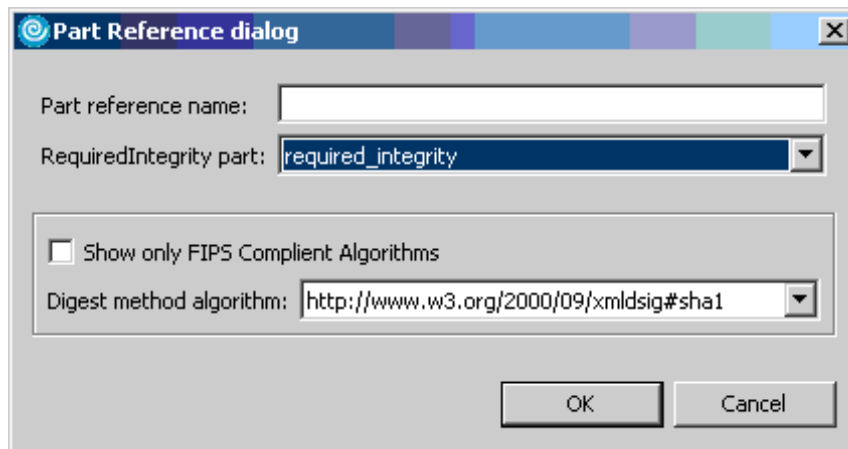


Figure 8-34 Part reference for signature validation

Expand the **Transforms** configuration section. Select the newly created **Part reference** and then add a new transform, as described in Figure 8-35. If a transform is not specified, errors will occur when the Web service response is received.

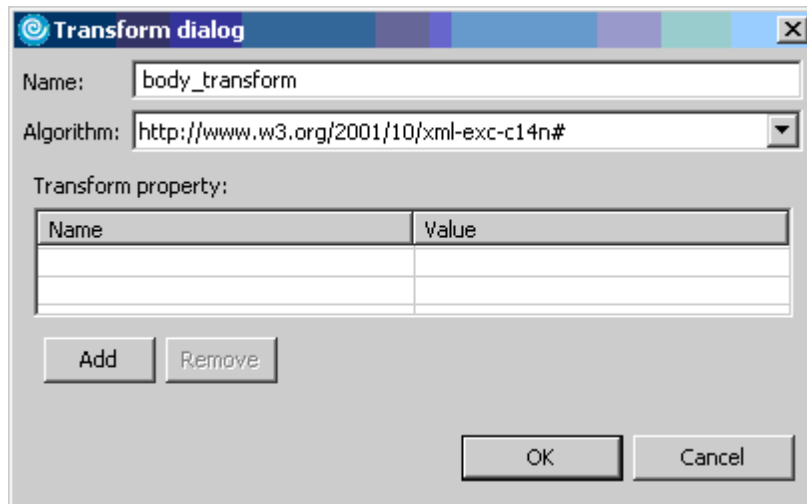


Figure 8-35 Transform for the body of the response after signature validation

Lastly, expand the **Encryption Information** configuration section. Add an entry, as shown in Figure 8-36 on page 223. The purpose of this entry is to link the confidentiality requirement defined on the WS Extension tab of the deployment descriptor and the information about the cryptographic keys that will be used to implement this requirement.

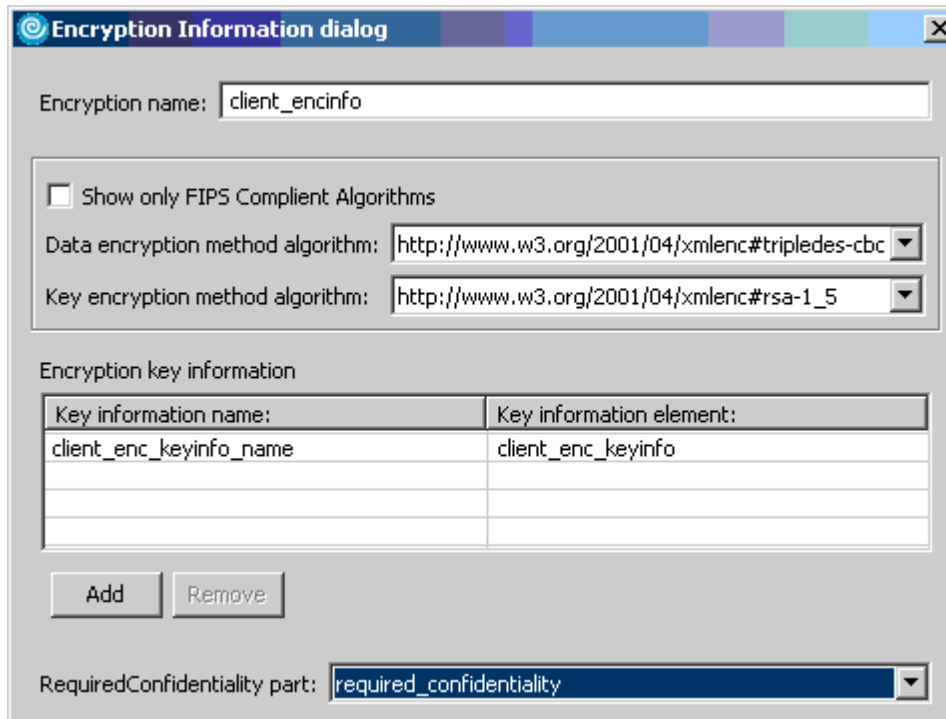


Figure 8-36 Configuration for how the response will be decrypted

The steps to enable the application client to process the secure Web service responses have been completed.

Save the deployment descriptor now. You may optionally close it.

8.2.4 Configure the application

With Rational Software Architect in the J2EE perspective, expand the **Web Services** folder in the Project Explorer view. Expand the **Services** folder and double-click the **AccountJCAService** Web service to open its deployment descriptor, as shown in Figure 8-37.

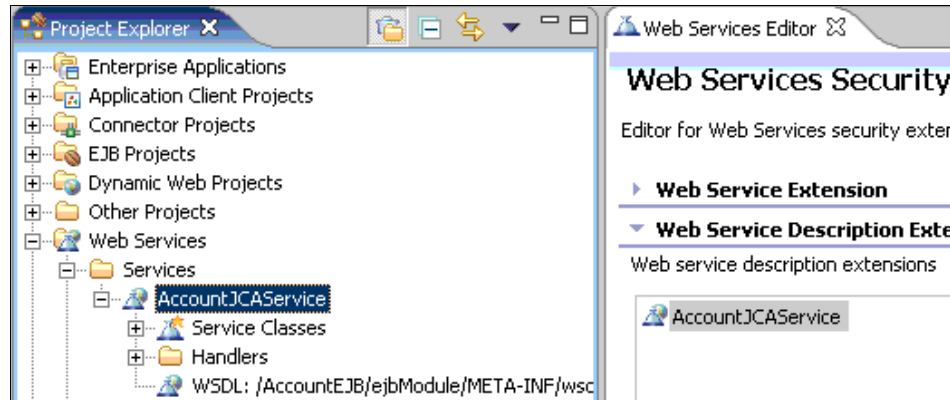


Figure 8-37 Web services editor

Request consumption

The application requires configuration to process the requests received from the application client:

- ▶ Decrypt the Web service request using application's private key.
- ▶ Validate the signature of the Web service request using the client's public key.
- ▶ Invoke the Federated Identity Manager trust service to:
 - Authenticate the user.
 - Authorize the user's access to the Web service.
 - Map the identity in the user name token to a SAML 2.0 assertion.
- ▶ Establish a login context for the user in the WebSphere Application Server, using JAAS.

Navigate to the Extensions tab in the Web services editor (Figure 8-38).

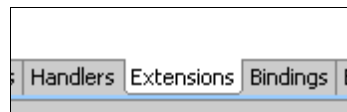


Figure 8-38 Extensions tab

The first thing to specify is the integrity requirements on the incoming messages. In this example, the client has signed the message body as well as the user name token provided in the WS-Security header. A shortcut is provided for specifying the body of the message, and is available from a drop-down menu in the Web services editor. The user name token cannot be simply referenced like that, so an XPath expression is needed to identify it. The XPath expression to be used is as follows:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' and local-name()='Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' and local-name()='UsernameToken']
```

Expand the **Port Component Binding** section, and select the **AccountJCA** entry.

Expand the **Request Consumer Service Configuration Details** configuration section. Expand the **Required Integrity** section and click **Add** to add an integrity requirement similar to what is shown in Figure 8-39.

For the body message part entry, the Message parts dialect value is available from a drop-down menu and should be:

```
http://www.ibm.com/websphere/webservices/wssecurity/dialect-was
```

For the message part entry described with the XPath expression, the *Message parts dialect* value is available from a pull-down menu and should be:

```
http://www.w3.org/TR/1999/REC-xpath-19991116
```

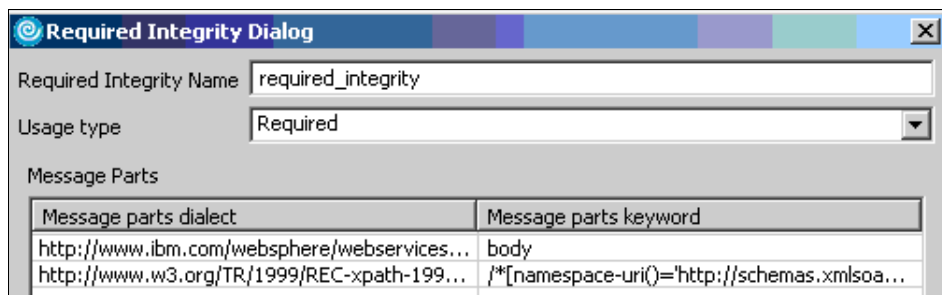


Figure 8-39 Specifying the integrity requirement

Expand the **Required Confidentiality** configuration section, and add a confidentiality entry to specify that the message body and the user name token should be encrypted.

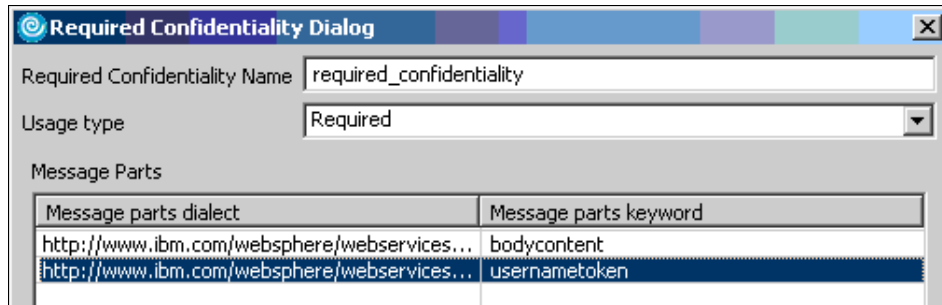


Figure 8-40 Specifying the confidentiality requirement

Navigate now to the **Binding Configurations** tab (Figure 8-41), which is where the abstract security requirements from the Extensions tab are associated with objects that can implement those requirements.

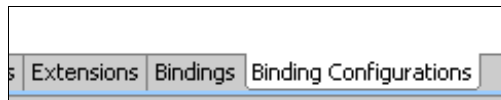


Figure 8-41 Binding Configurations tab

Expand the **Request Consumer Binding Configuration Details** section.

Expand the **Trust Anchor** configuration section and click **Add** to add a new trust anchor. As shown in Figure 8-42, the trust anchor identifies the key store to be used by the application.

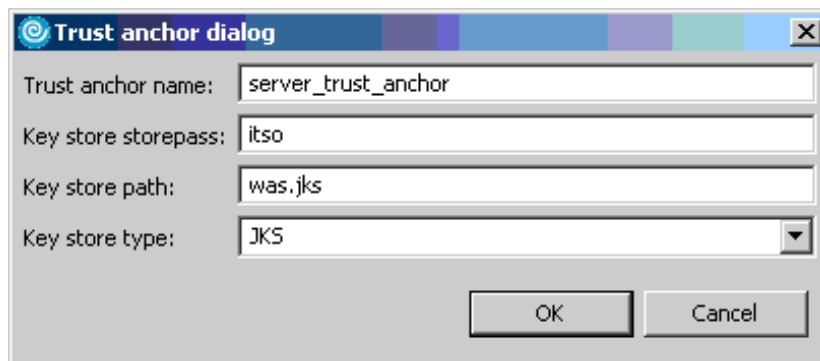


Figure 8-42 Trust anchor for request consumption

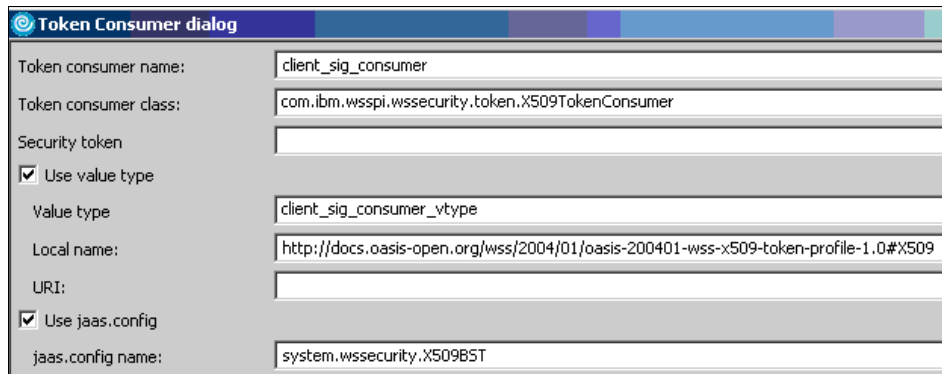
Expand the **Token Consumer** configuration section.

Two token consumers will be added to specify:

- ▶ The X.509 certificate that will be used to validate the signature on the incoming message (Figure 8-43 and Figure 8-44).
- ▶ The private key that will be used to decrypt the contents of the request body and the authentication token (Figure 8-45 on page 228 and Figure 8-46 on page 228).

For each of these two token consumers, two screen captures are provided here to capture the top and bottom of the *same* dialog box. It is important to apply the configuration in each screen capture; otherwise, errors will occur when the application tries to process Web service requests.

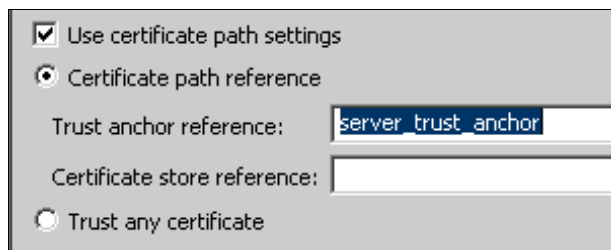
Once again, to pick the correct value for the Local name parameter, select a Value type of **X509 Certificate Token** before overwriting with the value shown in Figure 8-43.



The screenshot shows the 'Token Consumer dialog' window with the following fields and values:

Token consumer name:	client_sig_consumer
Token consumer class:	com.ibm.wsspi.wssecurity.token.X509TokenConsumer
Security token	
<input checked="" type="checkbox"/> Use value type	
Value type	client_sig_consumer_vtype
Local name:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
URI:	
<input checked="" type="checkbox"/> Use jaas.config	
jaas.config name:	system.wssecurity.X509BST

Figure 8-43 Token consumer for signature validation (top section of dialog)



The screenshot shows the bottom section of the dialog with the following options and values:

<input checked="" type="checkbox"/> Use certificate path settings	
<input checked="" type="radio"/> Certificate path reference	
Trust anchor reference:	server_trust_anchor
Certificate store reference:	
<input type="radio"/> Trust any certificate	

Figure 8-44 Token consumer for signature validation (bottom section of dialog)

The screenshot shows the 'Token Consumer dialog' with the following fields and values:

- Token consumer name: server_enc_consumer
- Token consumer class: com.ibm.wsspi.wssecurity.token.X509TokenConsumer
- Security token: (empty)
- Use value type
- Value type: server_enc_consumer_vtype
- Local name: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
- URI: (empty)
- Use jaas.config
- jaas.config name: system.wssecurity.X509BST

Figure 8-45 Token consumer for decryption (top section of dialog)

The screenshot shows the bottom section of the dialog with the following options and values:

- Use certificate path settings
- Certificate path reference
- Trust anchor reference: server_trust_anchor
- Certificate store reference: (empty)
- Trust any certificate

Figure 8-46 Token consumer for decryption (bottom section of dialog)

A Key locator configuration corresponding to the token consumers above is required. Expand the **Key Locators** configuration section and add entries for the client_sig_locator (Figure 8-47) and the server_enc_keylocator (Figure 8-48 on page 229).

The screenshot shows the 'Key Locator dialog' with the following fields and values:

- Key locator name: client_sig_keylocator
- Key locator class: com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator
- Use key store
- Key store storepass: (empty)
- Key store path: (empty)
- Key store type: (empty)

Figure 8-47 Locator for signature validation key

Key Locator dialog

Key locator name:

Key locator class:

Use key store

Key store storepass:

Key store path:

Key store type:

Key:

Alias	Key pass	Key name
was	itso	cn=was,o=itso,c=US

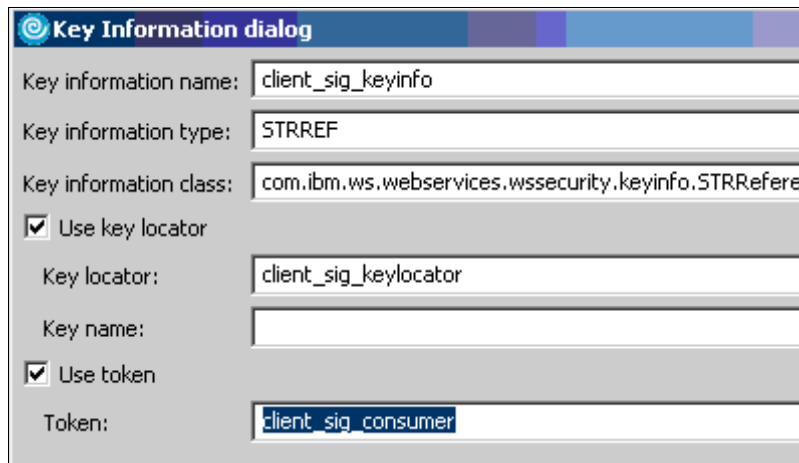
Figure 8-48 Locator for decryption key

A key information configuration links a token consumer with the key locator.

Expand the **Key Information** configuration section. Add a new entry for the client's signature validation, as shown in Figure 8-49. Because a client's public key is used for validation, the key information type is a security token reference.

Notes:

- ▶ The Key information class parameter will be automatically populated once the Key information type is selected.
- ▶ The Key locator and Token values should be available in the respective drop-down menus. If they are not, a prior configuration step is likely to have been missed.



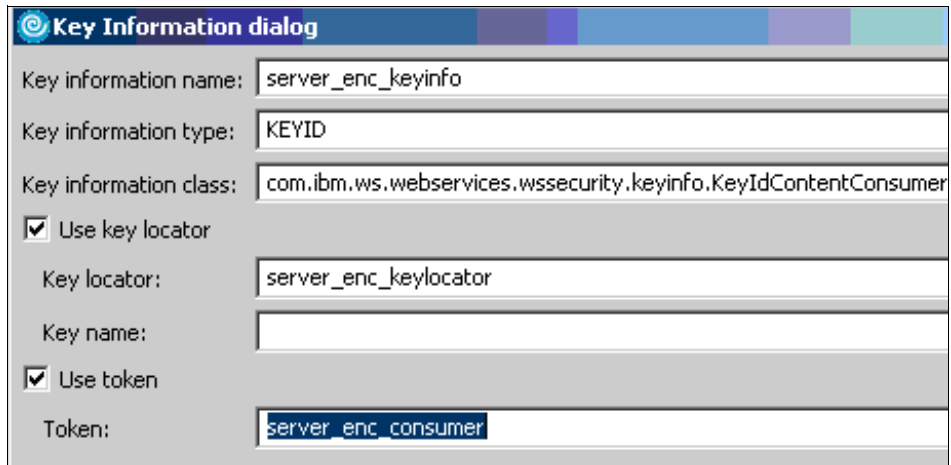
Key information name:	client_sig_keyinfo
Key information type:	STRREF
Key information class:	com.ibm.ws.webservices.wssecurity.keyinfo.STRRefere
<input checked="" type="checkbox"/> Use key locator	
Key locator:	client_sig_keylocator
Key name:	
<input checked="" type="checkbox"/> Use token	
Token:	client_sig_consumer

Figure 8-49 Key information for the validation of client signatures

Add a new key information entry for the server's decryption of the Web service request, as shown in Figure 8-50 on page 231. Because the server's private key is used for decryption, the key information type is a key identifier.

Notes:

- ▶ The Key information class parameter will be automatically populated once the Key information type is selected.
- ▶ The Key locator and Token values should be available in the respective drop-down menus. If they are not, a prior configuration step is likely to have been missed.



Key Information dialog

Key information name: server_enc_keyinfo

Key information type: KEYID

Key information class: com.ibm.ws.webservices.wssecurity.keyinfo.KeyIdContentConsumer

Use key locator

Key locator: server_enc_keylocator

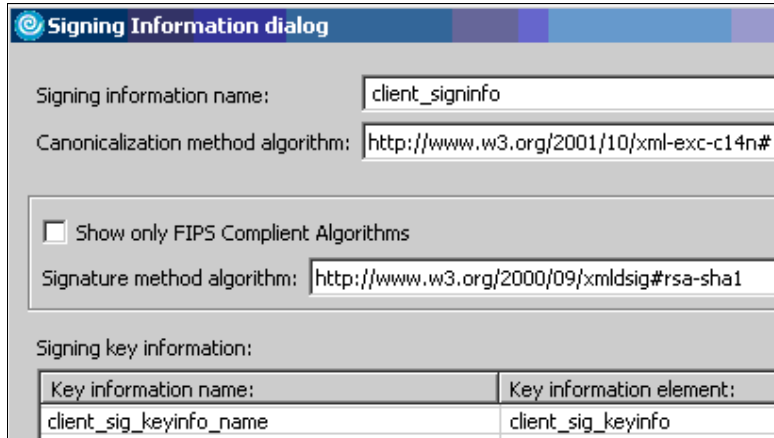
Key name:

Use token

Token: server_enc_consumer

Figure 8-50 Key information for decryption of Web service requests

Expand the **Signing Information** configuration section, and add a new entry similar to the one in Figure 8-51. This configuration item and the next two bind the cryptographic key to the integrity requirement defined earlier on the Extensions tab.



Signing Information dialog

Signing information name: client_signinfo

Canonicalization method algorithm: http://www.w3.org/2001/10/xml-exc-c14n#

Show only FIPS Compliant Algorithms

Signature method algorithm: http://www.w3.org/2000/09/xmldsig#rsa-sha1

Signing key information:

Key information name:	Key information element:
client_sig_keyinfo_name	client_sig_keyinfo

Figure 8-51 Signing information for Web service requests

With the newly created signing information selected, expand the **Part References** configuration section, and add a new entry, specifying the **Required Integrity** part (Figure 8-52). The `required_integrity` entry should be available in the drop-down menu. If it is not, it is likely that the integrity requirement was not correctly created on the Extensions tab.

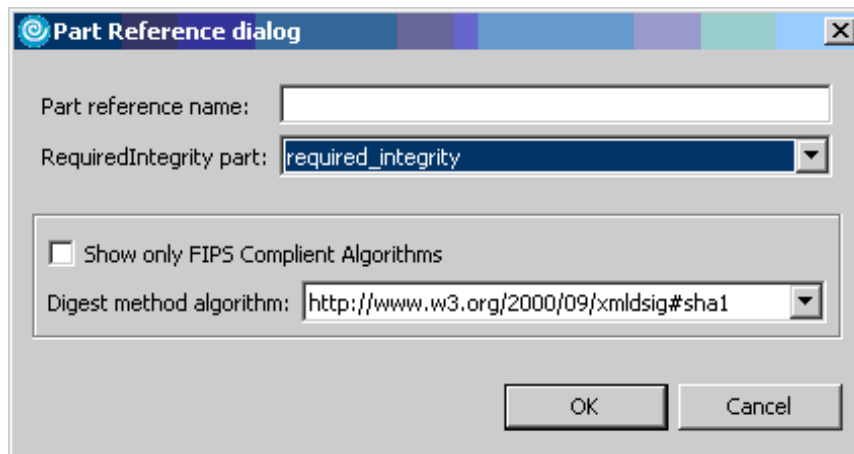


Figure 8-52 Part reference for integrity checks

With the newly created part reference selected, expand the **Transforms** configuration section and add a new entry (Figure 8-53). Only the name of the transform needs to be specified.

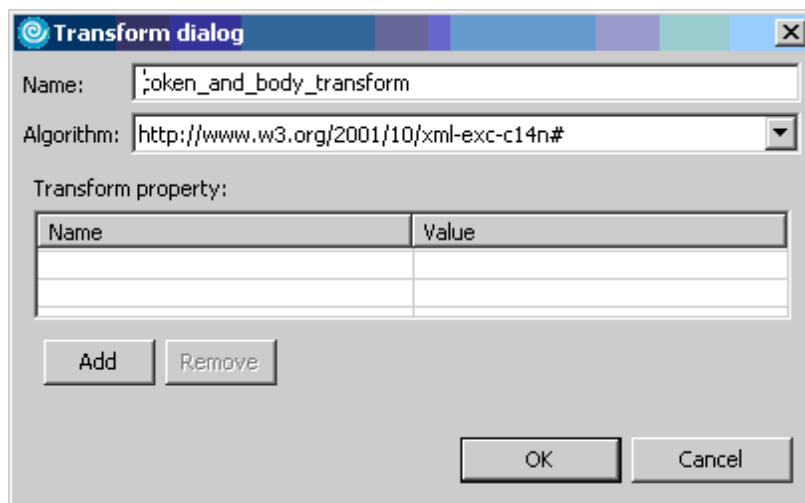


Figure 8-53 Transform configuration for signature validation of Web service requests

Expand the **Encryption Information** configuration section. Add a new entry, as shown in Figure 8-54, to bind the server's private key for decryption to the requirement for message confidentiality. The entry for the Required Confidentiality part field should be available in the drop-down menu. If it is not, it was not correctly created on the Extensions tab described by Figure 8-40 on page 226.

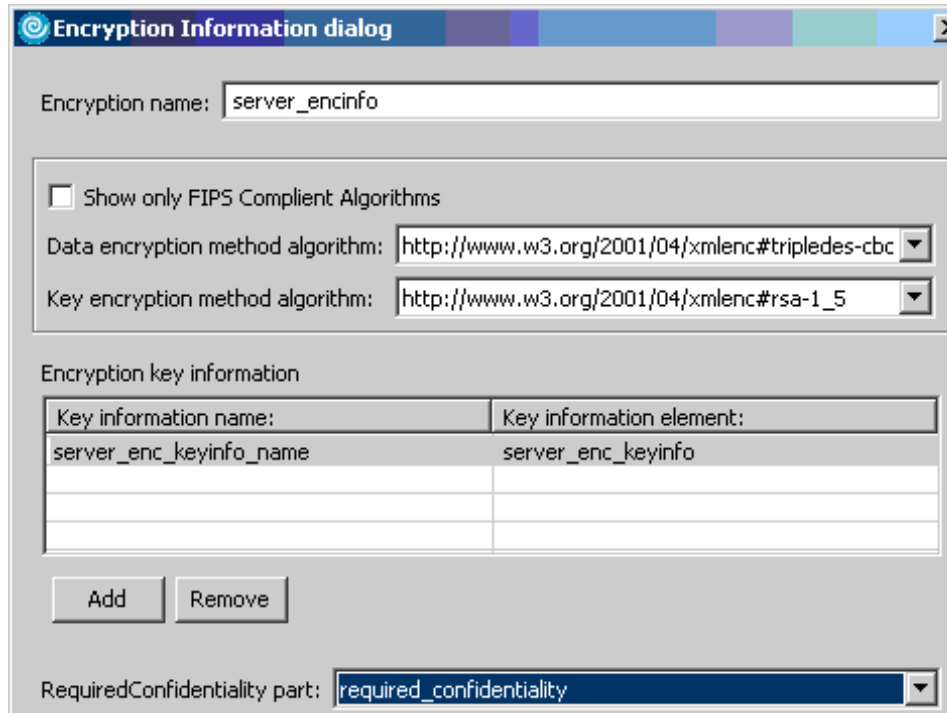


Figure 8-54 information about message decryption

The configuration performed for consuming requests thus far has been to configure aspects related to message protection, to be able to detect unauthorized modifications, and provide message privacy. The final configuration steps for request processing are related to authentication and authorization of users making the requests.

From the security requirements being implemented, Web service requests are expected to contain a user name token to identify the user. Federated Identity Manager will be used to:

- ▶ Authenticate the user based on the user name and password in the user name token.
- ▶ Map the identity in the incoming token to an internal identity.

- ▶ Authorize the user's ability to use the Web service based on their internal identity.
- ▶ Return the resulting identity (in the form of a SAML 2.0 assertion) to the WebSphere Application Server.

Return to the Extensions tab, and expand the **Required Security Token** configuration section. Add a new entry to specify that a user name token is expected (Figure 8-55). After selecting the **Token type**, the value of the Local name entry should be automatically populated. Also ensure that the Usage type is set to Required.

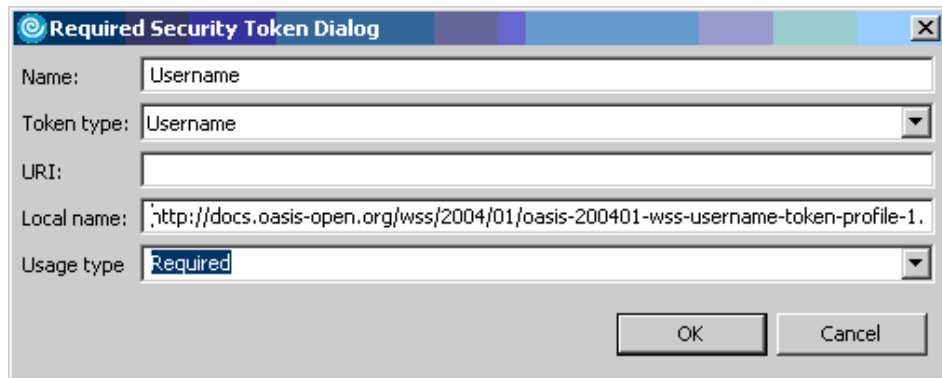


Figure 8-55 Specifying the security token expected in Web service requests

Now expand the **Caller Part** configuration section. Add an entry named SAMLA for the SAML 2.0 assertion that will be returned by Federated Identity Manager (see Figure 8-56 on page 235).

Use the following value for the URI parameter:

urn:oasis:names:tc:SAML:2.0:assertion

Do not click **OK** until the properties at the bottom of this dialog are also completed (see Figure 8-57 on page 235).

Figure 8-56 Configuration of SAML assertion as returned token type (top of dialog)

Scroll down to the bottom of this same dialog, and add two properties (Table 8-2), as shown in Figure 8-57.

Table 8-2 Caller part dialog property values

Property	Value
com.ibm.wsspi.wssecurity.caller.tokenConsumerNS	<leave blank>
com.ibm.wsspi.wssecurity.tokenConsumerLN	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

Figure 8-57 Configuration of SAML assertion as returned token type (bottom of dialog)

Return to the Binding Configurations tab, and locate the Token Consumer configuration section. Entries named `client_sig_consumer` and `server_enc_consumer` should already be present. Add a third entry, as shown in the next two screen shots (Figure 8-58 and Figure 8-59 on page 237).

The required token consumer class is from Federated Identity Manager, and is not present in the standard list, and must be typed in manually. The correct value is:

```
com.tivoli.am.fim.wssm.tokenconsumers.WSSMTokenConsumer
```

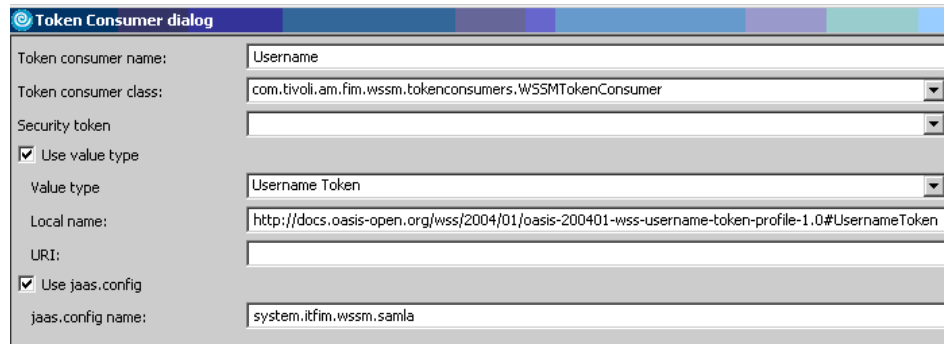


Figure 8-58 Token consumer configuration for the user name token (top of dialog)

Scroll down to the bottom of the dialog and add the following properties listed in Table 8-3.

Table 8-3 Token consumer property values

Property	Value
trust.service.call	true
trust.service.url	http://localhost:9081/TrustServer/SecurityTokenService
default.issuer.uri	bankpartner

The value for the `trust.service.url` parameter will vary according to the physical deployment of the solution components. In production deployments, channel level security, for example, the HTTPS protocol, may also be used.

The value of the `default.issuer.uri` property will need to match the value in the trust chain configuration described in 8.4, “Configure Web Service Security Management” on page 259.

Property:	
Name	Value
trust.service.call	true
trust.service.url	http://localhost:9080/TrustServer/SecurityTokenService

Figure 8-59 Token consumer configuration for the user name token (bottom of dialog)

This completes the configuration of the Web service request consumption.

Save the deployment descriptor now.

Response generation

In this section, the following security constraints are added to the application client:

- ▶ Sign Web service response with the application's private key.
- ▶ Encrypt Web service response with the client's public key.

Navigate to the Extensions tab of the deployment descriptor in the Web services editor. Expand the **Response Generator Service Configuration Details** configuration section.

Expand the **Integrity** configuration section. Add an entry to specify that the body of the Web service response should be protected from unauthorized modification (see Figure 8-60). Ensure that the value of the Order parameter is 1. The values for the properties in the Message Parts section of the dialog are available from the drop-down menus and do not have to be entered manually.

Integrity Dialog	
Integrity Name	BODY
Order	1
Message Parts	
Message parts dialect	Message parts keyword
http://www.ibm.com/websphere/webservices...	body

Figure 8-60 Adding message integrity

Expand the **Confidentiality** configuration section. Add an entry to specify that the content of the body of the Web service response should be encrypted (see Figure 8-61). Ensure that the value of the Order parameter is 2, so that this occurs after the message signing. The values for the properties in the Message Parts section of the dialog are available from the drop-down menus and do not have to be entered manually.

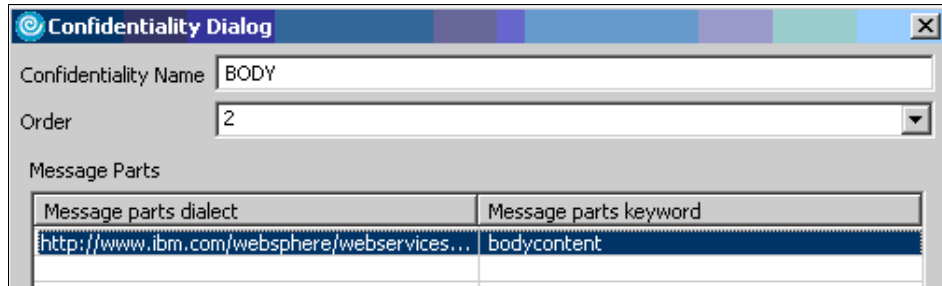


Figure 8-61 Adding message confidentiality

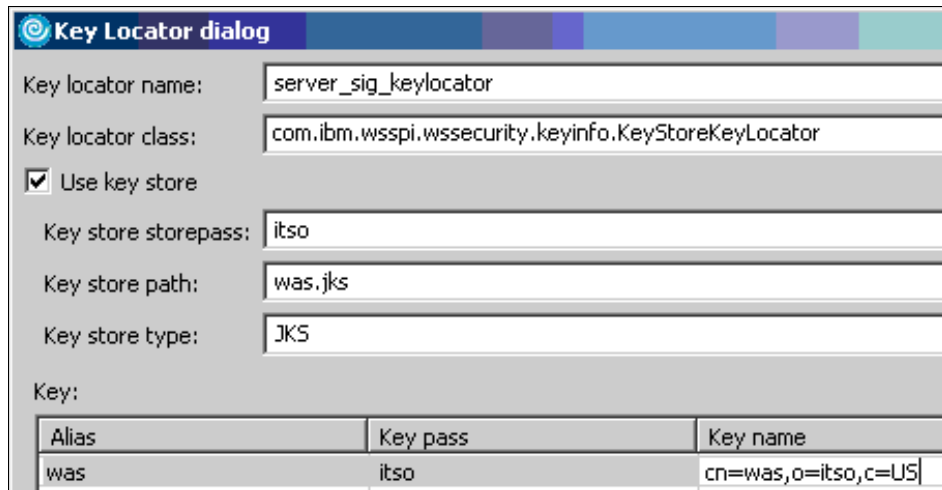
Navigate to the Binding Configurations tab. Expand the **Response Generator Binding Configuration Details** section.

Expand the **Token Generator** section, and add a new entry for the token used to sign the response (see Figure 8-62). The value for the Local name parameter can be automatically populated by choosing a Value type of X509 certificate token.

Alias	Key pass	Key name
was	itso	cn=was,o=itso,c=US

Figure 8-62 Token generator for the signing token

Expand the **Key Locators** configuration section. Add a new entry for the key locator for the signing key, as shown in Figure 8-63.



Key Locator dialog

Key locator name: server_sig_keylocator

Key locator class: com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator

Use key store

Key store storepass: itso

Key store path: was.jks

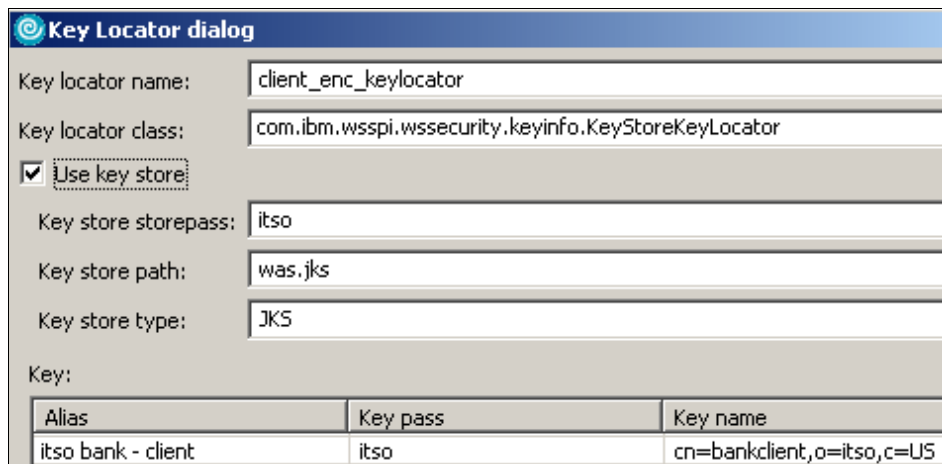
Key store type: JKS

Key:

Alias	Key pass	Key name
was	itso	cn=was,o=itso,c=US

Figure 8-63 Location of the signing key

Add a key locator for the certificate used to encrypt responses to the client, as shown in Figure 8-64.



Key Locator dialog

Key locator name: client_enc_keylocator

Key locator class: com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator

Use key store

Key store storepass: itso

Key store path: was.jks

Key store type: JKS

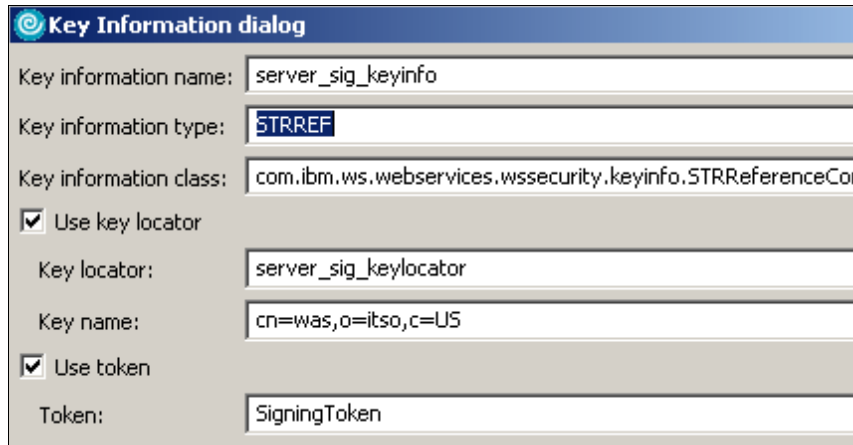
Key:

Alias	Key pass	Key name
itso bank - client	itso	cn=bankclient,o=itso,c=US

Figure 8-64 Location of the encryption key

Expand the **Key Information** configuration section.

Add a new entry for the signing key, as shown in Figure 8-65 on page 241. The value for the Key information type parameter should be STRREF.



Key Information dialog

Key information name: server_sig_keyinfo

Key information type: STRREF

Key information class: com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceCo

Use key locator

Key locator: server_sig_keylocator

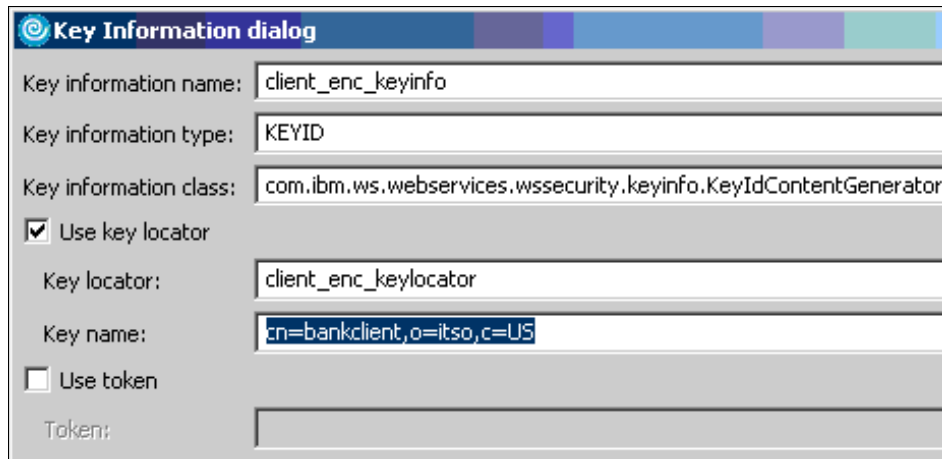
Key name: cn=was,o=itso,c=US

Use token

Token: SigningToken

Figure 8-65 Key information for the signing key

Add a key information entry for the encryption key (see Figure 8-66). The value for the Key information type parameter should be KEYID.



Key Information dialog

Key information name: client_enc_keyinfo

Key information type: KEYID

Key information class: com.ibm.ws.webservices.wssecurity.keyinfo.KeyIdContentGenerator

Use key locator

Key locator: client_enc_keylocator

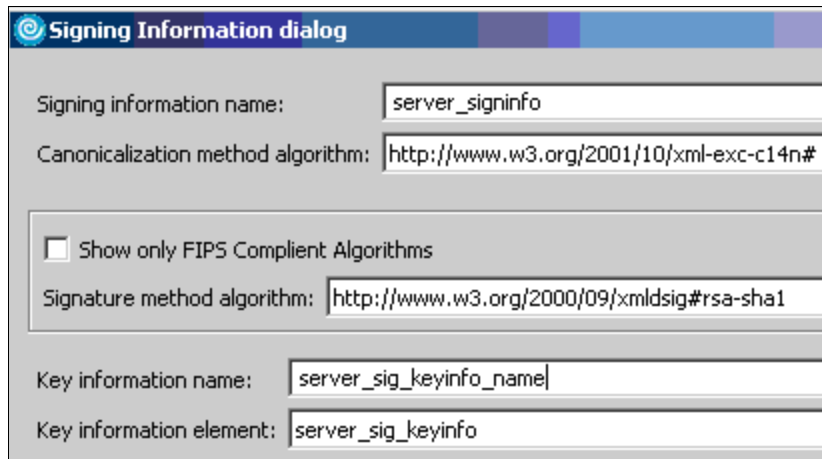
Key name: cn=bankclient,o=itso,c=US

Use token

Token:

Figure 8-66 Key information for the encryption key

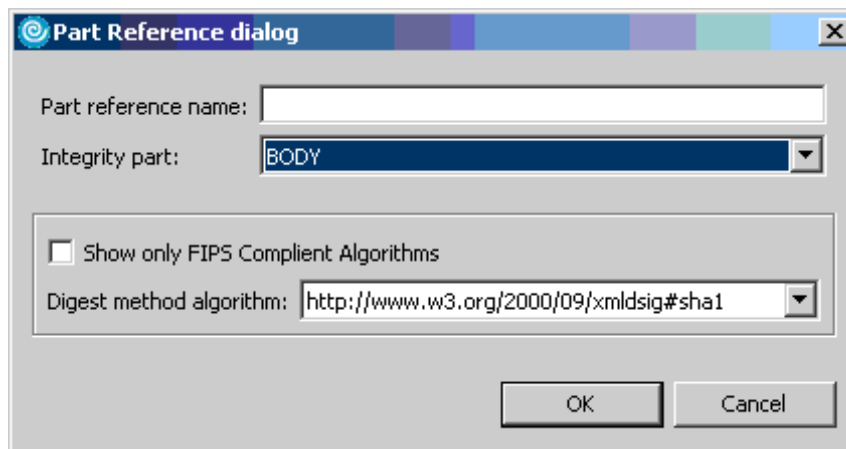
Expand the **Signing Information** configuration section. Add a new entry, as shown in Figure 8-67.



The image shows a dialog box titled "Signing Information dialog". It contains several input fields and a checkbox. The "Signing information name" field is filled with "server_signinfo". The "Canonicalization method algorithm" field is filled with "http://www.w3.org/2001/10/xml-exc-c14n#". There is a checkbox labeled "Show only FIPS Compliant Algorithms" which is currently unchecked. The "Signature method algorithm" field is filled with "http://www.w3.org/2000/09/xmldsig#rsa-sha1". The "Key information name" field is filled with "server_sig_keyinfo_name". The "Key information element" field is filled with "server_sig_keyinfo".

Figure 8-67 Signing information for Web service responses

Expand the **Part References** configuration section. Add a part reference, which binds to the BODY integrity requirement created earlier on the Extensions tab (see Figure 8-68). If the BODY integrity part is not available in the drop-down list, it should be added on the Extensions tab, as shown in Figure 8-54 on page 233 before continuing.



The image shows a dialog box titled "Part Reference dialog". It contains several input fields and a dropdown menu. The "Part reference name" field is empty. The "Integrity part" dropdown menu is set to "BODY". There is a checkbox labeled "Show only FIPS Compliant Algorithms" which is currently unchecked. The "Digest method algorithm" dropdown menu is set to "http://www.w3.org/2000/09/xmldsig#sha1". At the bottom of the dialog, there are "OK" and "Cancel" buttons.

Figure 8-68 Part reference to specify which part of the response will be signed

Expand the **Transforms** configuration section. Add a transform, as shown in Figure 8-69 on page 243, to complete the configuration for message integrity.

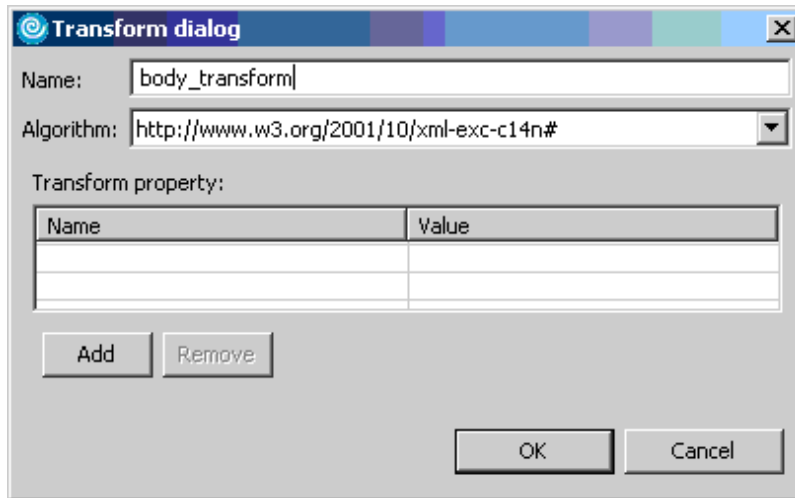


Figure 8-69 Transform for the response body when signed

Expand the **Encryption Information** configuration section. Add a new entry to complete the configuration for message confidentiality (see Figure 8-70).

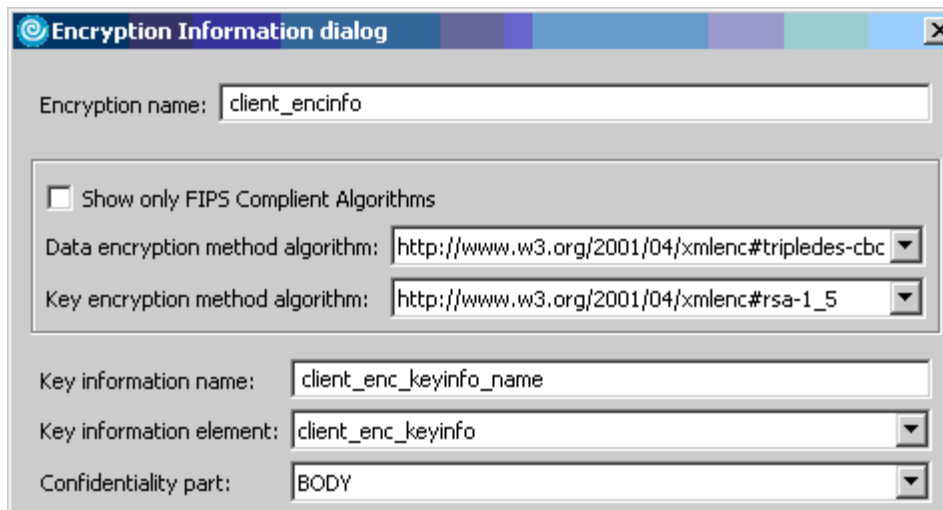


Figure 8-70 Encryption information for Web service responses

Specifying security configuration for Web service responses is now complete.

Save the deployment descriptor for the Web service.

8.2.5 Export the application with security configuration

The application and application client now have security configuration added. The updated versions of these applications need to be exported from Rational Software Architect so that they can be deployed in a WebSphere Application Server environment.

To export an enterprise application from Rational Software Architect, the steps are:

1. Locate the application in the J2EE perspective.
2. Right-click the application name and select **Export...** → **EAR file**.
3. Select a destination location for the exported file.
4. Optionally, export the source files of the application.

These steps need to be executed for the ITSOBanker2006 and ITSOBankerClient2006 applications.

8.3 Deploying the ITSO Banking Application

The steps in this section might typically be executed by an application administrator, who is responsible for deploying applications into WebSphere Application Server.

This deployment section assumes the following:

- ▶ IBM Tivoli Directory Server V6.0 is installed and one instance is configured.
- ▶ WebSphere Application Server V6.0.2.7 is already installed and at least one profile is created to host all the applications required.
- ▶ WebSphere Application Server Global Security is enabled and Tivoli Directory Server V6.0 is used as user registry.
- ▶ IBM Tivoli Access Manager V6.0 FP 3 is installed and the following components are configured:
 - Policy Server
 - Authorization Server
 - Runtime for Java
 - Web Portal Manager

- ▶ IBM Tivoli Federated Identity Manager V6.1 is installed and the following components are already configured:
 - Integrated Solution Console
 - Runtime and Management Services
 - Web Service Security Management
- ▶ Common Auditing and Reporting Service V6.0.1 is installed and configured.

8.3.1 Installing the CICS ECI resource adapter

The ITSO Banking Application requires a connection to a CICS Transaction Gateway using a *Java Connector Architecture* (JCA) resource adapter. Resource adapters are installed into application servers by adding the contents of a *Resource Adapter Module* (represented by a file with an extension of .rar) to the application server. A RAR file contains a collection of JAR files and a deployment descriptor (ra.xml) that describes the deployment properties of the resource adapter.

This section assumes that the CICS Transaction Gateway is already installed on a z/OS machine and that the CICS resource adapter (cicseci.rar) shipped with the CICS Transaction Gateway for z/OS is available.

Ensure that WebSphere Application Server is started, log in to the WebSphere Application Server Administrative Console, and navigate to **Resource** → **Resource Adapter**. Keeping the scope at the node level, click **Install RAR**, as depicted in Figure 8-71.

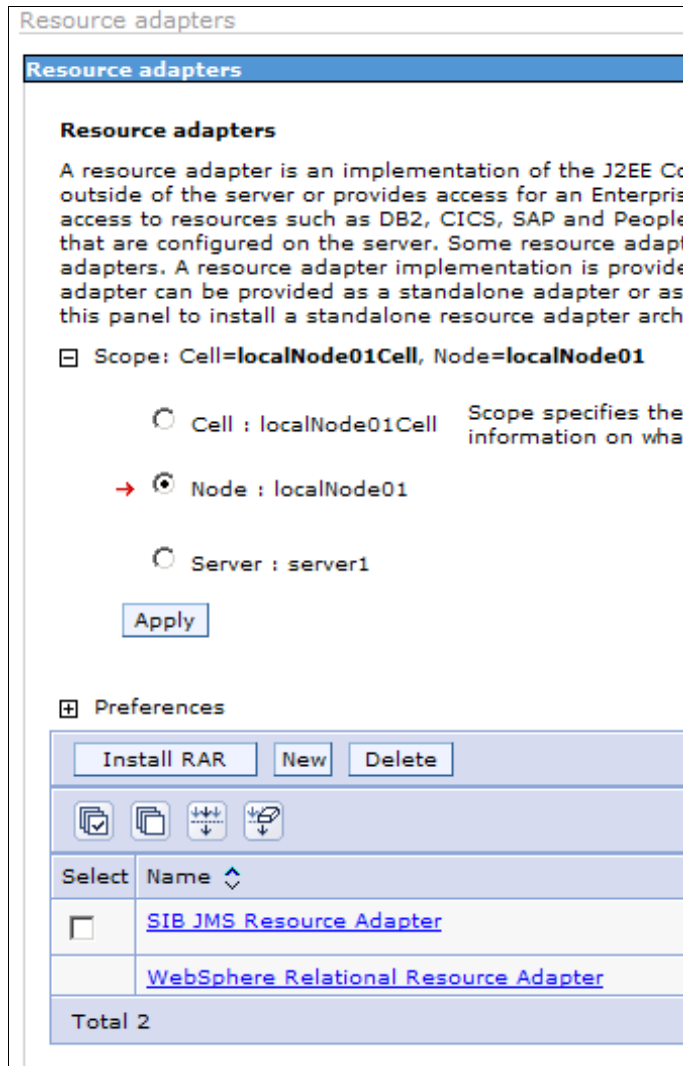


Figure 8-71 Installing a Resource Adapter rar file

Click **Browse** to navigate to the location of the cicsecei.rar file and select it. Click **Next** to continue, as shown in Figure 8-72.

Resource adapters

Install RAR File

RAR files can be installed using two methods. You can choose:

Path

Local path:

Specify path
c:\sts-jaas\cicsecei.rar

Server path:

Specify path

Scope

Node
localNode01

Figure 8-72 Specifying the rar file location

Leave all of the General Properties fields blank and click **OK**, as shown in Figure 8-73.

The image shows a dialog box titled "General Properties". It contains the following fields and controls:

- * Scope**: A text field containing the value "cells:localNode01Cell:nodes:localNode01".
- Name**: An empty text field.
- Description**: A text area with scroll bars, currently empty.
- Archive path**: An empty text field.
- Class path**: A text area with scroll bars, currently empty.
- Native path**: A text area with scroll bars, currently empty.

At the bottom of the dialog are three buttons: "OK", "Reset", and "Cancel".

Figure 8-73 Resource Adapter General Properties

When prompted, save the configuration to finalize the installation of the resource adapter.

8.3.2 Configuring the CICS Connection Factory

After installing a resource adapter, it is necessary to define a Connection Factory so that an enterprise application can use the resource adapter. In this section, a CICS Connection® Factory is defined. It will be used by the EJB within the ITSO Banking Application to connect to CICS.

The following CICS Transaction Gateway configuration information is required:

- ▶ ConnectionURL
- ▶ ServerName
- ▶ PortNumber
- ▶ TPName

If you have not closed the browser window, you should still be in the **Resource** → **Resource Adapter** view. If not, navigate through the WebSphere Application Server Administration Console to reach this point.

Click the newly created **ECIRResourceAdapter** link in order to display the configuration properties page (see Figure 8-74).

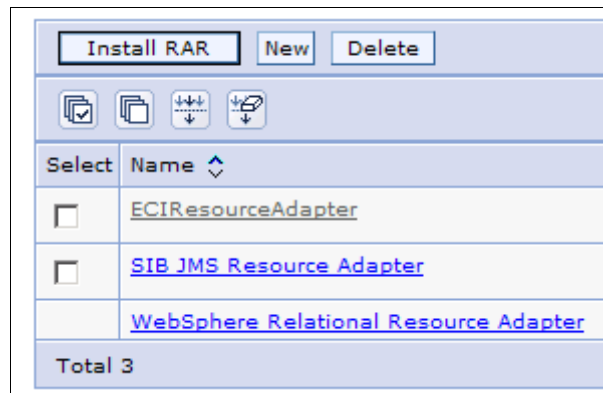


Figure 8-74 List of Resource Adapters available

On the configuration properties page, under the Additional Properties section, click **J2C connection factories**, as shown in Figure 8-75.



Figure 8-75 Resource Adapter additional properties

Click **New** and then fill in only the Name and JNDI name fields, as shown in Figure 8-76 on page 251. Use the values provided in Table 8-4.

Table 8-4 CICS Connection Factory properties

Parameter	Value
Name	CICS Connection Factory
JNDI name	eis/cicsConnectionFactory

Configuration

General Properties

* Scope

* Name

JNDI name

Description

* Connection factory interface

Category

Component-managed authentication alias

Component-managed authentication alias

Container-managed authentication

Container-managed authentication alias (deprecated in V6.0, use resource reference authentication settings instead)

Authentication preference (deprecated in V6.0, use resource reference authentication settings instead)

Mapping-configuration alias (deprecated in V6.0, use resource reference authentication settings instead)

Additional Properties

- [Connection pool properties](#)
- [Advanced connection factory properties](#)
- [Custom properties](#)

Related Items

Figure 8-76 Connection Factory configuration properties

Click **Apply**. Under Additional Properties, click the **Custom Properties** link.

At this point, specify the values for the CICS environment being used. In the ITSOBank environment, those custom properties are shown in Figure 8-77.

[Resource adapters](#) > [ECIResourceAdapter](#) > [J2C connection factories](#) > [CICS Connection Factory](#) > **Custom properties**

Custom properties that may be required for resource providers and resource factories. For example, most database vendors require additional custom properties for data sources that access the database.

Preferences

Name	Value	Description	Required
ServerName	SCSCPA2B	ServerName	false
ConnectionURL	tcp://wtcs58.itso.ibm.com	ConnectionURL	false
PortNumber	2006	PortNumber	false
UserName		UserName	false
Password		Password	false
ClientSecurity		ClientSecurity	false
ServerSecurity		ServerSecurity	false
KeyRingClass		KeyRingClass	false
KeyRingPassword		KeyRingPassword	false
TranName	DPLC	TranName	false
TPNName	DPLC	TPNName	false
TraceLevel	1	TraceLevel	false
Total 12			

Figure 8-77 Connection Factories custom properties

In this scenario, **UserName** and **Password** do not need to be specified, since that information will be obtained from the Federated Identity Manager Trust Service by a JAAS Login module.

Click the **Save** link when done to save these configuration changes to the master configuration.

8.3.3 Configure a JAAS login module

This section describe how to configure a JAAS login module to get a RACF Passticket from the Federated Identity Manager trust service. This login module will be used in conjunction with the resource adapter described in the previous section.

The ITSO Banking Application contains a JAAS login module in a jar file named soa-jaas-login.jar.

This jar file needs to be copied into the <WAS_HOME>/lib directory so that it can be used by WebSphere Application Server.

The commands used in the ITSOBank environment are shown in Example 8-1.

Example 8-1 Installing the JAAS login module

```
# cp /labstuff/soa-jaas-login.jar /opt/IBM/WebSphere/AppServer/lib/  
# chmod 644 /opt/IBM/WebSphere/AppServer/lib/soa-jaas-login.jar
```

Important: Restart WebSphere Application Server so that the new jar file is available.

Log in to the WebSphere Application Server Administrative Console, and navigate to **Security** → **Global Security**. In the Authentication section, expand **JAAS Configuration**. Click **Application Login** to display the current configuration. Click **New** to add a new configuration.

Provide a meaningful name for the alias, such as FIMforJCA, and click **Apply** in the dialog displayed in Figure 8-78.

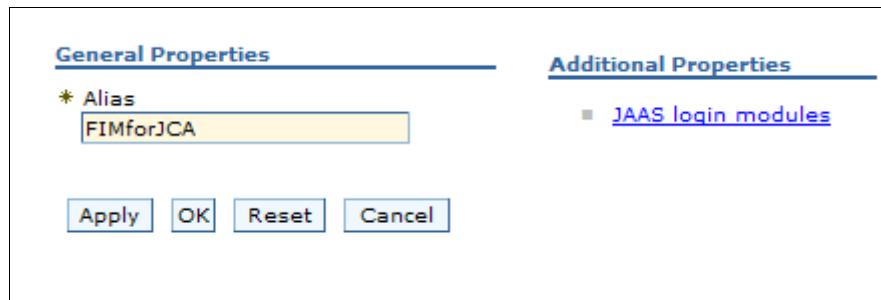


Figure 8-78 JAAS Application Login configuration

Under Additional Properties, click **JAAS login modules**. Click **New** to create a new JAAS login module (Figure 8-78 on page 253). Use the values in Table 8-5.

Table 8-5 JAAS login module configuration

Parameter	Value
module class name	com.tivoli.am.fim.jaas.login.loginmodules.FIMPrincipalMapper
use login module proxy	selected
authentication strategy	REQUIRED

The JAAS Login properties page should look like Figure 8-79. Click **Apply** to save this new configuration, but remain on this page.

Figure 8-79 JAAS login module properties

Under Additional Properties, click the **Custom properties** link and create the following properties, listed in Table 8-6, by clicking **New** for each one and entering the name and value. Modify the value of the stsendpoint parameter consistent with the Federated Identity Manager environment being used.

Table 8-6 Custom properties for the JAAS login module

Name	Value
appliesto	ITSOBanker2006
cache cleanup interval	30
cache expiration interval	60
issuer	fimprincipal

Name	Value
stsendpoint	http://local.demo.com:9080/TrustServer/SecurityTokenService

After doing this, the custom properties page should resemble Figure 8-80.

New Delete		
Select	Name	Value
<input type="checkbox"/>	appliesto	ITSOBanker2006
<input type="checkbox"/>	cache cleanup interval	30
<input type="checkbox"/>	cache expiration interval	60
<input type="checkbox"/>	delegate	com.tivoli.am.fim.jaas.login.loginmodules.FIMPrincipalMapper
<input type="checkbox"/>	issuer	fimprincipal
<input type="checkbox"/>	stsendpoint	http://local.demo.com:9080/TrustServer/SecurityTokenService
Total 6		

Figure 8-80 JAAS Login module custom properties

Click **Save** to complete the configuration of the JAAS Login Module.

8.3.4 Deploy the ITSO Banking Web service

In this section, the ITSO Banking Web service application will be installed into WebSphere Application Server.

The ITSO Banking Web service application uses WS-Security capabilities to secure the service. This requires cryptographic keys stored in a keyfile as specified in its key locator configuration entries (defined in the binding files that were created by development during the application assembly process in “Response consumption” on page 214).

A sample keystore, named was.jks, is supplied with this IBM Redbook. The keystore must be copied into the WebSphere Application Server server profile.

Note: Instructions on how to obtain the unsecured versions of the sample application are available in Appendix D, “Additional material” on page 389.

Assuming we have unzipped the additional material into the /labstuff subdirectory you have to use the following command to copy the file into the correct location:

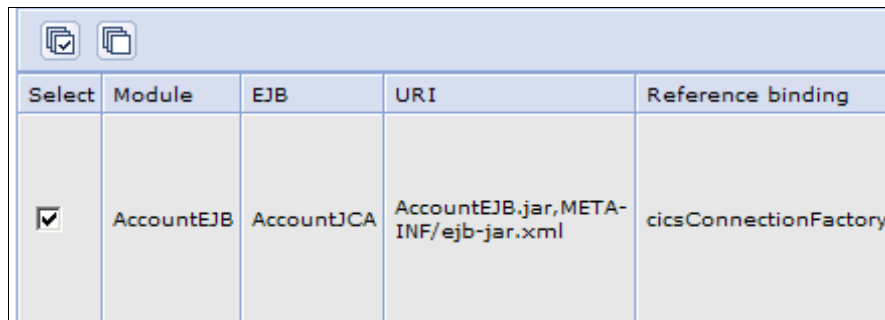
```
# cp /labstuff/was.jks /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/
```

Log on to the WebSphere Application Server Administrative Console. Navigate to **Enterprise Applications** → **Install New Application** to display the first screen of the application install process.

Click **Browse** and locate the ITSOBanker2006.ear file. Click **Next**. For the next four configuration pages, accept the default settings by clicking **Next** on each of those pages.

On step 5, Map Resource References to Resources, you need to specify an existing resource JNDI name, and a custom authentication method for the AccountEJB module.

Mark the check box for the AccountEJB module (on the bottom left of the page), as shown in Figure 8-81.



Select	Module	EJB	URI	Reference binding
<input checked="" type="checkbox"/>	AccountEJB	AccountJCA	AccountEJB.jar,META-INF/ejb-jar.xml	cicsConnectionFactory

Figure 8-81 Selecting the EJB module resource mapping

On the existing resource JNDI name scroll bar, select **eis/cicsConnectionFactory** and click **Apply**, as shown in Figure 8-82 on page 257.

To set multiple existing resource JNDI names:

1. Select one or more checkboxes in the table
2. Select existing resource JNDI name
3. Click Apply

Specify existing Resource JNDI name:

Figure 8-82 JNDI resource definition for the EJB module resource mapping

Select the **Use custom login configuration** radio button and select the **FIMforJCA** entry from the drop-down menu, as shown in Figure 8-83. Click **Apply**.

Specify authentication method:

none

Use default method

Select authentication data entry

Use custom login configuration

Select application login configuration

Figure 8-83 Custom application login definition for the EJB module resource mapping

With the authentication method now defined, a new button, Mapping Properties, is available on the bottom right of the page, as shown in Figure 8-84.

Login configuration

Resource authorization:
Container

Authentication method:
FIMforJCA

Figure 8-84 Login configuration mapping properties

Click the **Mapping Properties** button and define a custom property (Figure 8-85) using the values in Table 8-7.

Table 8-7 Custom property definition for ITSOBank2006 login configuration

Name	Value
appliesto	ITSOBanker2006

Figure 8-85 Custom properties for the application login configuration

Click **OK** twice. The bottom of the Map Resource Reference page should resemble Figure 8-86.

Figure 8-86 Login and JNDI definition configuration results

Click **Next** to continue deploying the application. Accept the defaults for the remaining configuration steps and then click **Finish**.

Save the changes to the master configuration to end the deployment of the application.

8.4 Configure Web Service Security Management

Federated Identity Manager *Web Service Security Management* (WSSM) provides components that reside on WebSphere Application Server and integrate with WebSphere Application Server's WS-Security token processing. In particular, the Federated Identity Manager WSSM provides WebSphere Application Server components for:

- ▶ Web service providers: For inbound security tokens via a WebSphere Application Server Token Consumer
- ▶ Web service clients: For outbound security tokens via a WebSphere Application Server Token Generator

Both the WSSM token consumer and token generator contain a trust client and contact the Federated Identity Manager trust service to provide token management facilities for WebSphere Application Server Web service applications.

In this section, only the service provider token consumer feature is used. The assumption is that a Federated Identity Manager domain is already configured and available.

8.4.1 Configure the WSSM trust chains

The trust service processing for WSSM requires two trust chains that execute together:

- ▶ A WSSM partner chain
- ▶ A WSSM application chain

The *partner chain* is created by using the Federated Identity Manager ISC console while the *application chain* is created automatically after running the `wsd12tfim` utility using the WSDL file of the service.

Configuring the WSSM partner chain

Before configuring the partner trust chain, we need to define a user name module instance, since this is the token type that the application expects to receive from the clients.

Open a browser to the Federated Identity Manager ISC console and log in as the `iscadmin` user. Navigate to **Configure Trust Service** → **Module Instances**.

Click the **Create** button to display a list of the installed and available trust service modules, as shown in Figure 8-87.

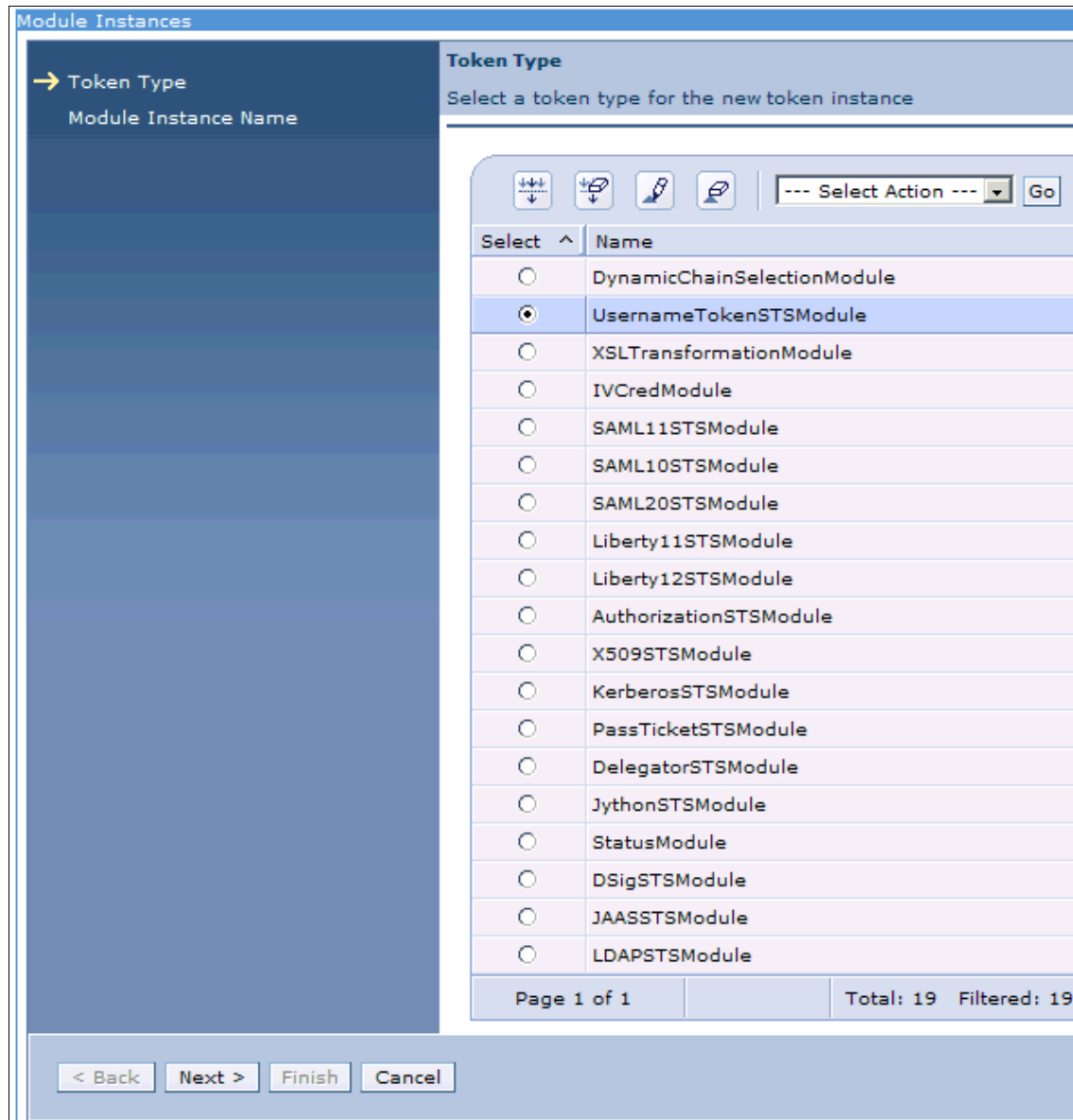
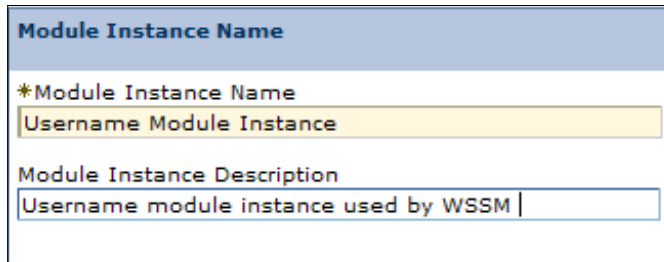


Figure 8-87 Token Type Modules in Federated Identity Manager trust service

Select **UsernameTokenSTSMModule**, and click **Next** to proceed to the module instance detail page.

Provide a name for the instance, such as Username Module Instance, and, optionally, a description, and click **Finish** (see Figure 8-88).



The screenshot shows a form titled "Module Instance Name" with a blue header. Below the header, there are two input fields. The first field is labeled "*Module Instance Name" and contains the text "Username Module Instance". The second field is labeled "Module Instance Description" and contains the text "Username module instance used by WSSM |".

Figure 8-88 Module Instance definition

Ignore the message prompting for the restart of WebSphere Application Server at this time.

Tip: The restart of WebSphere Application Server is not required until all new configuration steps are complete and ready to be used by the Federated Identity Manager runtime.

After a user name token instance is configured, it is possible to configure a WSSM partner chain.

Still using the Federated Identity Manager Console, navigate to **Configure Web Services Security** → **Partners**.

Click **Create** to display this first window of the WSSM Partner Wizard, as depicted in Figure 8-89.

Figure 8-89 Web Service Security Partner definition

WSSM partners create a module chain that is matched using a regular expression string compared against the URL of the target Web service. The RequestSecurityToken message from the WSSM token consumer module on WebSphere Application Server will contain the full URL in the AppliesTo element; however, it is possible to use the regular expression pattern matching capabilities of the trust service to match on just the path portion of the endpoint URL, making it independent of the host name or IP address used to access the service.

Enter a regular expression string that will match the path portion of the Web service address (including the leading “.”), and the Company Name, as described in Table 8-8.

Table 8-8 Partner contact information

Field name	Value
Web Service URL	*/routerProject/services/AccountJCA
Company Name	ITSO Bank

Click **Next**. Select the token type received from your partner. Select the Username Module Instance created in the previous section. Click **Next**.

The next step is to configure options specific to the user name token type.

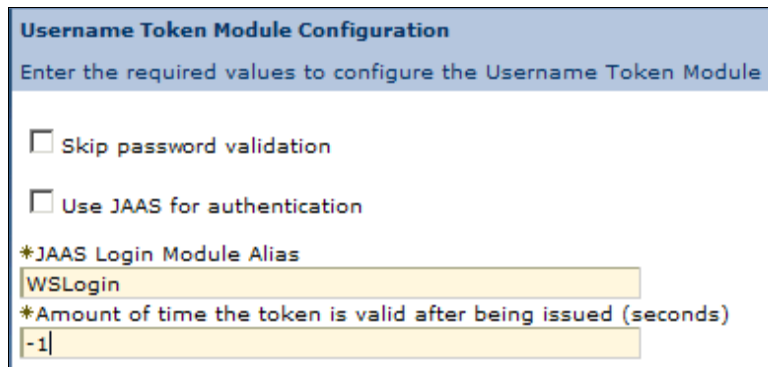
The mechanism used for validation of the user name and password can be one of these options:

- ▶ No password validation
- ▶ Use a JAAS login module
- ▶ Use Access Manager (using the Access Manager user registry)

It may be that the identity contained in the user name token submitted by the partner does not exist in the Access Manager registry, but validation logic is still required. In that case, a JAAS login module can be used.

In this scenario, Access Manager will be used to validate the user name token, so leave the **Skip password validation** and **Use JAAS for authentication** check boxes deselected.

Set the **Amount of time the token is valid after being issued** parameter to -1 indicating that the token is valid indefinitely, as shown in Figure 8-90. This will remove time synchronization as an issue in this prototype environment, but consideration should be given in a production environment to setting this to a low, non-negative value, such as 5 minutes to reduce the risk of replay attacks.



The screenshot shows a configuration form titled "Username Token Module Configuration". Below the title is a subtitle: "Enter the required values to configure the Username Token Module". There are three main sections: 1) Two checkboxes, "Skip password validation" and "Use JAAS for authentication", both of which are unchecked. 2) A section labeled "*JAAS Login Module Alias" with a text input field containing "WSLogin". 3) A section labeled "*Amount of time the token is valid after being issued (seconds)" with a text input field containing "-1".

Figure 8-90 Security Token Identity Mapping

Click **Next** to proceed to the page where an identity mapping rule can be defined. Both the partner chain and Web service application chain can contain identity mapping logic to process the identity of the token submitted by the Web service client. Mapping rules are expressed using XML stylesheets (XSLT).

Note: In this case, an identity mapping rule is not required, since the user will enter their ITSO bank user ID and password.

To provide a better understanding of how identity mapping works in WSSM, a one-to-one mapping rule is demonstrated. Some additional identity attributes are also added.

Figure 8-91 on page 265 shows the mapping rule that will be used. Two fixed value attributes named `company_name` and `is_racf_user` are added to the user identity. Remember, these attributes are added for demonstration purposes only, and are not used in authorization decisions in this implementation.


```

<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" />
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>
  <xsl:template match="//stsuuser:AttributeList">
</xsl:template>
<xsl:template match="//stsuuser:Principal">
  <stsuuser:Principal>
    <stsuuser:Attribute name="name">
      <stsuuser:Value><xsl:value-of
select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuus
er:Value" /></stsuuser:Value>
    </stsuuser:Attribute>
    <stsuuser:Attribute name="issuer"
type="urn:oasis:names:tc:SAML:2.0:assertion">
      <stsuuser:Value>http://local.demo.com</stsuuser:Value>
    </stsuuser:Attribute>
  </stsuuser:Principal>
  <stsuuser:AttributeList>
    <stsuuser:Attribute name="company_name" type="wsm_example">
      <stsuuser:Value>ITSOBank</stsuuser:Value>
    </stsuuser:Attribute>
    <stsuuser:Attribute name="is_racf_user" type="wsm_example">
      <stsuuser:Value>TRUE</stsuuser:Value>
    </stsuuser:Attribute>
  </stsuuser:AttributeList>
</xsl:template>
</xsl:stylesheet>

```

Figure 8-91 XSLT mapping rule

Click **Next** and review the summary of the partner chain configuration. Click **Finish** to save this configuration.

A Restart WebSphere message will again be displayed by the Federated Identity Manager Console. Use the **Dismiss** button again and continue with the next section.

Create a WSSM Web service application trust chain

The configuration of the WSSM Web service application trust chain is created using the **wsdl2tfim.sh** utility. The utility uses this general syntax:

```
$(WSSM_INSTALL)/wsdl2tfim.sh -t <token-type-module-name> <path-to-wsdl>
```

The <token-type-module-name> parameter specifies the token type desired to be returned by the trust service.

The <path-to-wsdl> parameter specifies a file system path to the WSDL file for the Web service.

Note: `wsdl2tfim.sh` must be executed on a machine that has the WebSphere Application Server and Federated Identity Manager WSSM libraries installed, but it *does not* have to be the machine that is hosting the Web service application.

Note: If Federated Identity Manager WSSM and WebSphere Application Server are not installed in their default locations, edit `wsdl2tfim.sh` and update the values of the `JAVA_HOME`, `WAS_HOME`, and `FIM_HOME` environment variables accordingly.

The `wsdl2tfim.sh` utility uses a properties file, `wsdl2tfim.properties`, for configuration parameters. The values in `wsdl2tfim.properties` must be updated with values corresponding to the location and configuration of the Federated Identity Manager management application.

In the ITSOBank environment, there is no WebSphere Application Server cluster, WebSphere Application Server global security is enabled, and no WebSphere Application Server SSL client authentication is used.

The `wsdl2tfim.properties` is located in the `wssm` directory of your Federated Identity Manager installation. In the ITSOBank environment, its content is shown in Example 8-2.

Example 8-2 wsdl2tfim.properties files in ITSO environment

```
# The name of the FIM domain whose STS configuration is to be updated
```

```
fimDomainName=ITSO-Domain
```

```
# The name of the server where the FIM management service is running
```

```
serverName=local.demo.com
```

```
# The SOAP connector port of the server where the FIM management
service is running (usually 8880)

serverPort=8880

# The name of the WAS application server (eg. server1) - only required
in a non-clustered configuration

appServerName=server1

.....

# A WAS user ID that's in the WAS administrator group

adminName=wasadmin

# The password for the WAS user ID

adminPassword=passw0rd

# The WAS trust keystore file name - refer to the SSL repertoire
configuration in WAS

trustedKeystorePath=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/etc/
DummyServerTrustFile.jks

# The WAS trust keystore password - refer to the SSL repertoire
configuration in WAS.
# A password is only required if the trust keystore also contains
private keys.

trustedKeystorePassword=WebAS
```

Note: If secure communication (over HTTPS) to the Federated Identity Manager trust service is used, you need to update the `wssm.properties` file for information related to secure communication.

Before running the `wSDL2fim.sh` script, it is necessary to have a copy of the application WSDL file. It is shipped as part of the Web service application. From the Federated Identity Manager WSSM directory, run the `wSDL2fim.sh` script, as shown in Example 8-3 on page 268.

Example 8-3 Invoking wsd12tfim.sh

```
# cd /opt/IBM/FIM/wssm
# ./wsd12tfim.sh -t SAML20Module
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/localNode0
1Cell/ITSOBanker2006.ear/routerProject.war/wsd1/ejbs/AccountJCA.wsd1
```

This command specifies that a SAML token containing a SAML 2.0 assertion should be used to return the mapped identity to the ITSO Banking application.

The wsd12tfim.sh utility contacts the Federated Identity Manager Management application to create an application trust chain for the one <port> contained in the input WSDL.

Note: In order to verify that a new application trust chain has been created, log in to the Federated Identity Manager console and browse to **Configure Trust Service** → **Trust Service Chains** and it should show the new chain, as shown in Figure 8-92 on page 268.

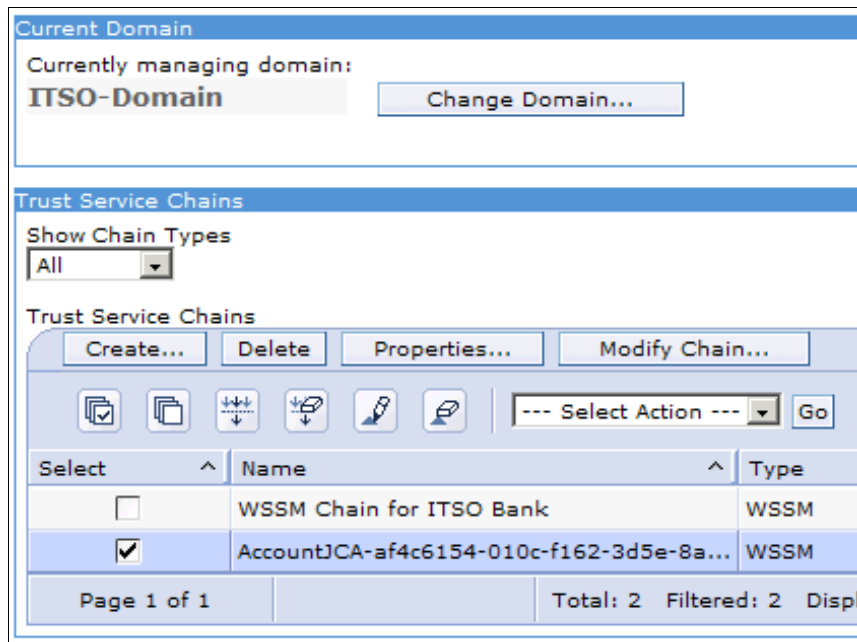


Figure 8-92 WSSM application trust chain

Configure WSSM Web service application trust chain

The application trust chain is created by the `wsd12tfim.sh` utility, but further configuration using the trust chain editor in the Federated Identity Manager console is required.

The logic that performs the transformation, along with any identity and attribute manipulation, is contained within a mapping module. Federated Identity Manager 6.1 allows either a standard mapping module that processes XSLT transformation rules, or a custom mapping module (that must be written using the trust service APIs).

In the ITSOBank environment, an XSLT transformation mapping module in the partner chain is used. No further mapping rule is needed in the application trust chain for the ITSOBank environment.

After selecting the application trust chain, as shown in Figure 8-92 on page 268, click the **Properties** button.

Set the **Issuer Address** to urn:itfim:wssm:tokenconsumer (Figure 8-93).

Lookup Type

Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenType)
 Use XPath to Define Custom Lookup Rule

AppliesTo

Address
REGEXP:(.*/routerProject/services/AccountJCA)

Service Name
:

Port Type
:

Issuer

Address
urn:itfim:wssm:tokenconsumer

Service Name
:

Port Type
:

Token Type

Figure 8-93 Trust service chain properties

Specifying urn:itfim:wssm:tokenconsumer for the Issuer ensures that this trust chain can be used for token exchange messages from the WSSM token consumer.

Click **OK** to save the changes. Dismiss the message prompting for WebSphere Application Server to be restarted.

Again, select the application chain, but this time click **Modify Chain**. A warning message is displayed, similar to the one in Figure 8-94.



Figure 8-94 Application generated chain modification warning

Click **Continue With Modification**.

In the ITSOBank environment, Access Manager is to be used for Web service authorization. To enable this feature, it is necessary to modify the authorization module that is the second module instance in the application chain.

Select it as shown in Figure 8-95. Click the **Properties** button.

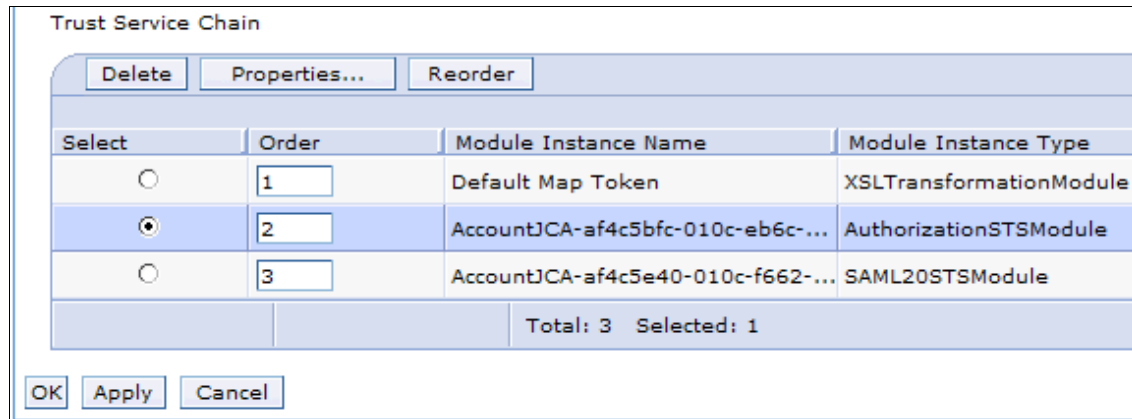


Figure 8-95 Application trust chain

Set the value of the **Web service protected object name** parameter to /itfim-wssm/wssm-default/ItsoBankWSDL/AccountJCAService, as shown in Figure 8-96.

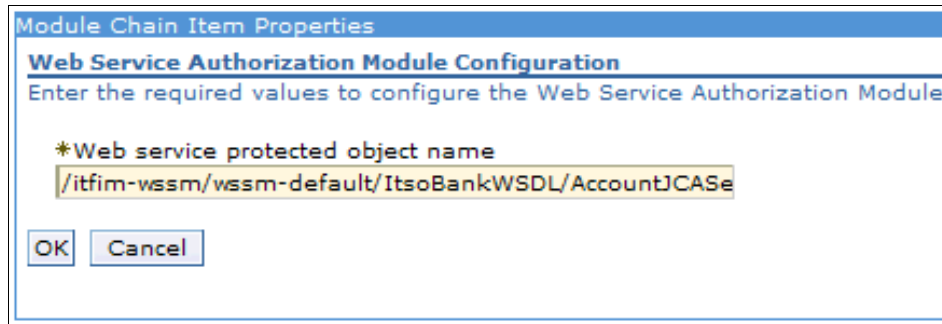


Figure 8-96 Access Manager object space definition into the Web Service Authorization module

This path represents the absolute path of the base portion of the Access Manager object space used for the Access Manager authorization check. This must match what the **wsd12tam** utility (to be used later) will produce.

Click **OK** to return to the module instance list. Select the **SAML20STSModule** instance. Click **Properties**.

This module is present in the chain because we used option **-t SAML20Module** with the **wsd12tfim.sh** utility. It is used to return the identity to the WSSM token consumer. The parameters on the screen in Figure 8-97 on page 273 control how the SAML 2.0 token will be created.

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

*Number of seconds before the issue date that an assertion is considered valid

*Amount of time the assertion is valid after being issued (seconds)

Include the following attribute types (a "*" means include all types)

Enable the Signing of Assertions

Select Key for Signing Assertions

Select the key for encrypting assertion elements for this partner

Encrypt assertions (an Encryption key is required)

Encrypt assertion Attribute elements (an Encryption key is required)

Encrypt NameID elements in assertions (an Encryption key is required)

Block Encryption Algorithm

Figure 8-97 SAML module instance configuration for the application chain

Provide an organization name (for example, ITSOBank) and include an "*" for the attributes mapping in the SAML assertion, indicating that all attributes should be included in the outgoing SAML token.

The WSSM JAAS login module that will process the returned SAML assertion assumes that the SAML token is from a trusted source (the Federated Identity Manager trust service, to be precise) and does not validate the signature or

validity period of the SAML assertion. So, signing the SAML assertion in this scenario is not needed. Deselect the **Enable the Signing of Assertions** check box.

Click the **OK** button.

This time, select to restart WebSphere Application Server.

Web service authorization using Access Manager

Federated Identity Manager V6.1 WSSM provides an additional, but optional, capability to the Web services security support in WebSphere Application Server that is the authorization of Web service access using a Tivoli Access Manager policy.

The default structure of the Access Manager protected object space provides a logical representation of the Web service without reference to how the Web service is invoked or where the Web service is physically installed, as shown in Figure 8-98.

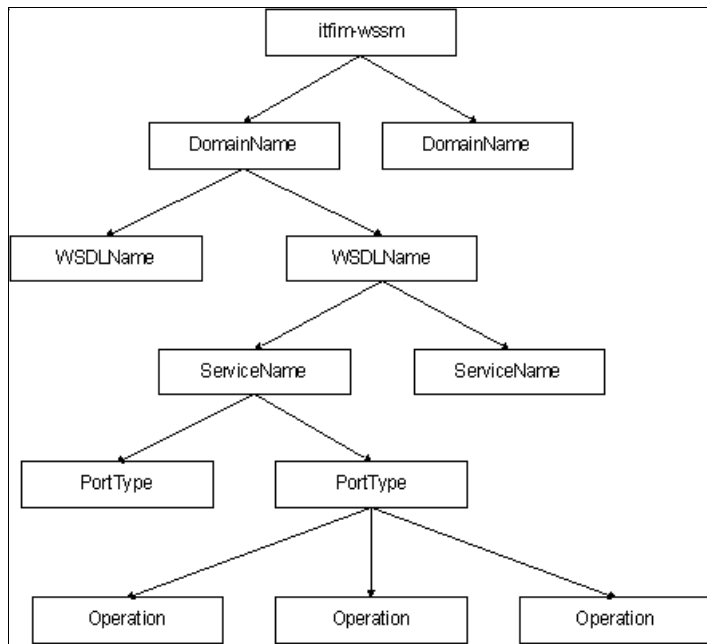


Figure 8-98 WSSM default Access Manager object model

The object space structure includes:

- ▶ The root object (itfim-wssm). This name is used to create a separate Access Manager object space that separates Web services from other object spaces.
- ▶ A domain object may be used to partition the object space. For example, there may be separate domains for different lines of business. The default domain name used by wsdl2tam is wssm-default.
- ▶ A container object (labeled as WSDLName in Figure 8-98 on page 274) allows for grouping of related Web services that share a common access policy. The name of this object is arbitrary and is specified by the `-n <name>` operand of the `wsdl2tam.sh` utility. In our case, it is `ItsoBankWSDL`.
- ▶ The name of the service as specified in the service's WSDL document. In this case, the WSDL contains:

```
<wsdl:service name="AccountJCAService">
```

- ▶ The name of the service's portType, as specified in the service's WSDL document. In this scenario:

```
<wsdl:portType name="AccountJCA">
```

- ▶ An object for each of the service operations, as specified in service's WSDL document. In this scenario, it is:

```
<wsdl:operation name="getBalance">
```

Note: This default object space model does not contain any objects that represent the host, TCP/IP port, or URL path used to access the service over HTTP. The intent is for the protected object design to be transport neutral.

In order to perform an authorization decision, Access Manager needs the following information:

- ▶ Credential (who is requesting access)

A credential is an Access Manager credential created by the authorization trust module using the identity in the current `STSUniversalUser`.

In this scenario, since an exact one-to-one mapping rule is used, this identity is exactly the same identity as contained in the security token in the ITSO Banking Client request. This identity must be a valid user in the Access Manager registry.

- ▶ Resource (what is being accessed)

The resource will be the Access Manager protected object provided by the Federated Identity Manager authorization trust module instance as a consequence of the processing of the `RequestSecurityToken` message from the WSSM token consumer.

- ▶ Action (how the resource is being accessed)
The action will be [WebService]i - 'i' for invoke, as per the default configuration using the `wsdl2tam` utility.

Configure Web service authorization using `wsdl2tam` utility

The creation of the Access Manager commands for the modification of the Access Manager object space is done using the `wsdl2tam.sh` utility.

The `wsdl2tam.sh` utility uses the following operands:

- ▶ The `-n` option allows an arbitrary container level entry in the Access Manager object space to allow grouping of Web services that will share a common access policy. In our scenario, we used `ItsoBankWSDL`.
- ▶ The `-tam` option provides the output file name for the generated Access Manager commands.
- ▶ The path to the WSDL file.

Note: The `wsdl2tam.sh` utility requires access to various Java libraries shipped with Federated Identity Manager WSSM. In case you have installed the Federated Identity Manager WSSM in a path different from the default, you need to edit `wsdl2tam.sh` and change the value for the `FIM_HOME` variable.

Go to the WSSM home directory in a command shell and invoke the `wsdl2tam` utility, as shown in Example 8-4.

Example 8-4 Invoking `wsdl2tam.sh`

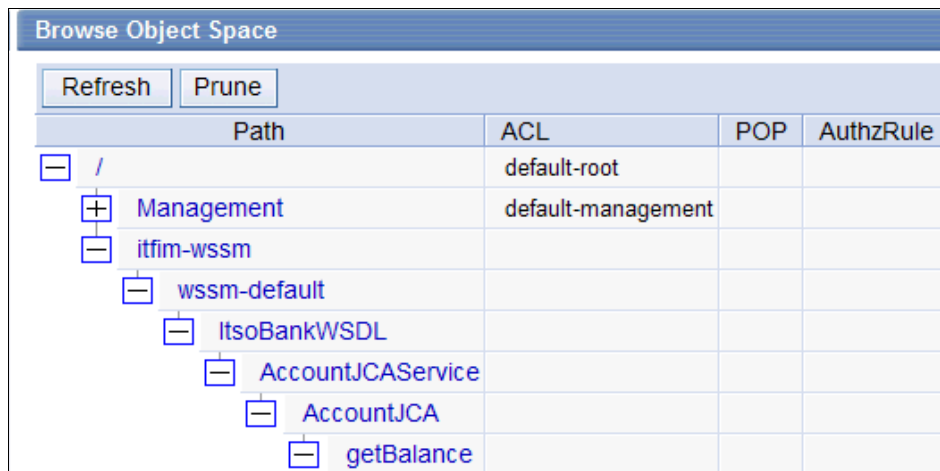
```
# cd /opt/IBM/FIM/wssm
# ./wsdl2tam.sh -n ItsoBankWSDL -tam tam.script
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/localNode0
1Cell/ITS0Banker2006.ear/routerProject.war/wsdl/ejbs/AccountJCA.wsdl
```

Upon successful completion, review the contents of the `tam.script` file. It should contain a sequence of `pdadmin` commands to create objects in the Access Manager object space for this Web service. Example 8-5 on page 277 shows how to apply these commands to the Access Manager environment.

Example 8-5 Updating the Access Manager object space using the output from wsd12tam

```
# pdadmin -a sec_master -p passw0rd tam.script
cmd> action group create WebService
cmd> action create i wsm 0 WebService
cmd> objectspace create /itfim-wsm wsm 0
cmd> object create /itfim-wsm/wsm-default wsm 0 ispolicyattachable
yes
cmd> object create /itfim-wsm/wsm-default/ItsoBankWSDL wsm 0
ispolicyattachable yes
cmd> object create
/itfim-wsm/wsm-default/ItsoBankWSDL/AccountJCAService wsm
0 ispolicyattachable yes
cmd> object create
/itfim-wsm/wsm-default/ItsoBankWSDL/AccountJCAService/AccountJCA
wsm 0 ispolicyattachable yes
cmd> object create
/itfim-wsm/wsm-default/ItsoBankWSDL/AccountJCAService/AccountJCA/getBalance
wsm 0 ispolicyattachable yes
```

The object space created can be examined using the Access Manager Web Portal Manager application, in our case, it is shown in Figure 8-99.



Path	ACL	POP	AuthzRule
/	default-root		
Management	default-management		
itfim-wsm			
wsm-default			
ItsoBankWSDL			
AccountJCAService			
AccountJCA			
getBalance			

Figure 8-99 Access Manager object space

For this example implementation, an Access Manager policy needs to be created that allows authorized users to invoke the *AccountJCAService*. A group named created ITSUsers is created. This group is added to a policy that is attached to

the protected object representing this service. Later, users can be added to the group to have access enabled for them.

From within the **pdadmin** utility, run commands shown in Example 8-6.

Example 8-6 Access Manager policy configuration for the ITSO Bank Web service

```
pdadmin sec_master> group create ITS0Users cn=ITS0Users,c=us ITS0Users
pdadmin sec_master> acl create ITS0Policy
pdadmin sec_master> acl modify ITS0Policy set group ITS0Users
T[WebService]i
pdadmin sec_master> acl attach
/itfim-wssm/wssm-default/ItsoBankWSDL/AccountJCAService/AccountJCA
ITS0Policy
```

Configuring a Federated Identity Manager trust chain

The ITSOBank application uses a customized JAAS configuration from the CICS adapter (see 8.3.2, “Configuring the CICS Connection Factory” on page 248 and 8.3.3, “Configure a JAAS login module” on page 253). The JAAS login module connects to the Federated Identity Manager trust service to retrieve a valid RACF user ID and Passticket on behalf of the user.

An additional trust chain will be defined for this purpose and an LDAP mapping module will be used to look up the RACF user ID for the incoming user ID. The RACF user ID is defined in an attribute of the LDAP user object.

In this section, we are going to deploy the LDAP mapping module and then define a new trust service chain that uses it.

A Federated Identity Manager trust service module is provided in form of a jar file, its related descriptor, and a resource bundle used for the creation of the new configuration panel into the Federated Identity Manager console.

The plug-in used for the ITSO environment is composed of:

- ▶ A `ldap-sts-plugin-module.jar` file that is the custom module.
- ▶ A `plugin.xml` file that describes the plug-in.
- ▶ An `itfim-ldap-map-i18n.jar` file that is the resource bundle needed by the Federated Identity Manager console.

In the `<FIM_HOME>/plugins` directory, create a new subdirectory named `com.tivoli.am.fim.ldap.plugin_1.0`. Copy the three files above into that new subdirectory.

Important: On a non-Windows machine, ensure that the new subdirectory has file permissions 770 and the files have permissions 660. The subdirectory and the files should be owned by root.

Copy the resource bundle `itfim-ldap-map-i18n.jar` to the `lib` directory of the Federated Identity Manager console WAR directory at:

```
<ISC_INSTALL_ROOT>/PortalServer/installedApps/FIMConsole_PA_1_0_9D.ear/  
PA_1_0_9D.war/WEB-INF/lib
```

Important: On a non-Windows machine, ensure that the file has permissions of 644.

The addition of a new plug-in equates to a new version of the Federated Identity Manager runtime application. In order to re-deploy the runtime through the Federated Identity Manager console, it is necessary to manually increment the `serialId` property in `<FIM_HOME>/pkg/software.properties`.

Increment the existing value for `com.tivoli.am.fim.rte.software.serialId`, and optionally update the `com.tivoli.am.fim.rte.software.displayName`.

In the ITSO environment, the values were changed, as in Example 8-7.

Example 8-7 Increment the software serialId

```
from :  
com.tivoli.am.fim.rte.software.serialId=1153289542454  
com.tivoli.am.fim.rte.software.displayName=6.1.0 [060524a]  
  
to:  
com.tivoli.am.fim.rte.software.serialId= 1153289542455  
com.tivoli.am.fim.rte.software.displayName=6.1.0 [060525-CST-LDAP-STS]
```

Close any browser windows using the Federated Identity Manager console and restart the ISC server so that the new change can take effect.

Log in to the Federated Identity Manager console and browse to **Domain Management** → **Runtime Node Management** and click **Deploy Runtime**, as shown in Figure 8-100.

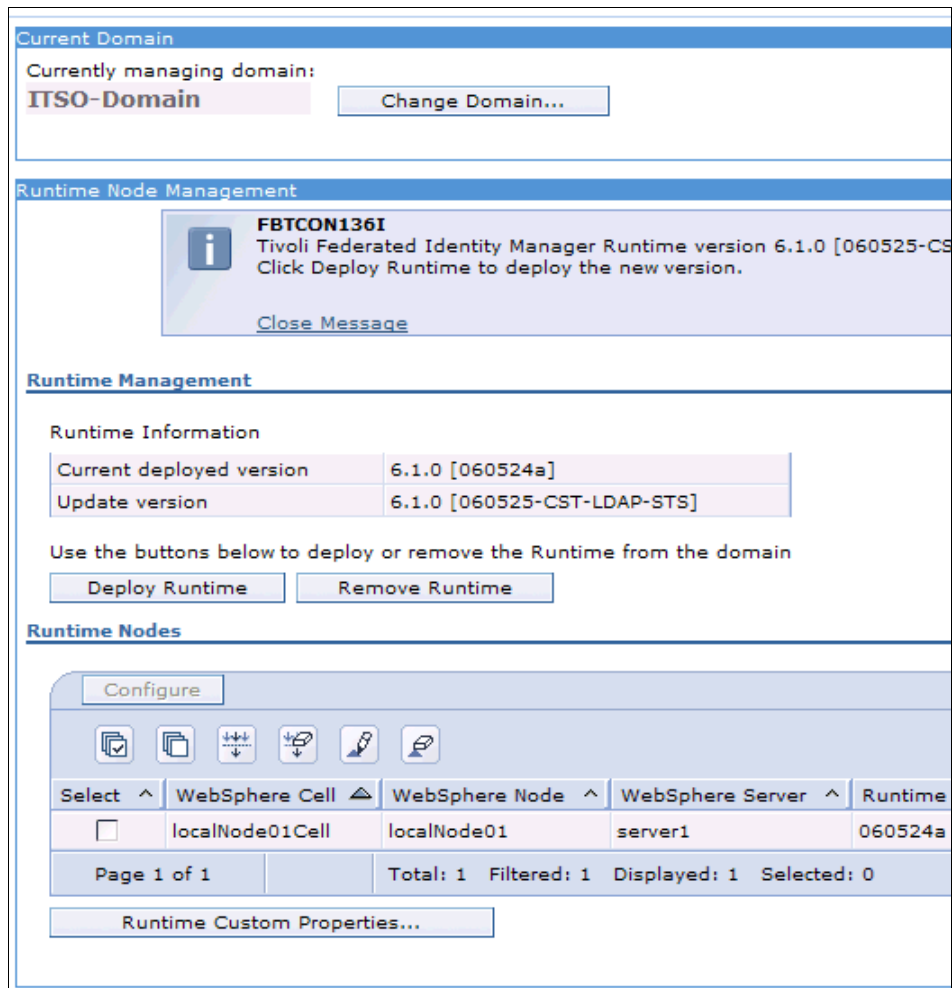


Figure 8-100 Runtime deployment

Restart WebSphere Application Server when prompted by the Federated Identity Manager console.

After the WebSphere Application Server has restarted, it is necessary to define an instance of the new LDAP module and an instance of the Passticket module.

In the Federated Identity Manager console, browse to **Configure Trust Service** → **Module Instances** and click **Create** to show the list of the token modules available, as shown in Figure 8-101.

Token Type
Select a token type for the new token instance

--- Select Action --- Go

Select ^	Name ^
<input type="radio"/>	DynamicChainSelectionModule
<input type="radio"/>	UsernameTokenSTSMModule
<input type="radio"/>	XSLTransformationModule
<input type="radio"/>	IVCredModule
<input type="radio"/>	SAML11STSMModule
<input type="radio"/>	SAML10STSMModule
<input type="radio"/>	SAML20STSMModule
<input type="radio"/>	Liberty11STSMModule
<input type="radio"/>	Liberty12STSMModule
<input type="radio"/>	AuthorizationSTSMModule
<input type="radio"/>	X509STSMModule
<input type="radio"/>	KerberosSTSMModule
<input type="radio"/>	PassTicketSTSMModule
<input type="radio"/>	DelegatorSTSMModule
<input type="radio"/>	JythonSTSMModule
<input type="radio"/>	StatusModule
<input type="radio"/>	DSigSTSMModule
<input type="radio"/>	JAASSTSMModule
<input checked="" type="radio"/>	LDAPSTSMModule

Page 1 of 1 Total: 19 Filtered: 19 Displayed: 19 Select

Figure 8-101 Token type module list

Select **LDAPSTSTModule** and click **Next** to specify a name and a description for this new instance, as shown in Figure 8-102.

The screenshot shows a configuration window titled "Module Instance Name". It has two input fields. The first field, labeled "*Module Instance Name", contains the text "LDAP Mapping Token". The second field, labeled "Module Instance Description", contains the text "A Custom LDAP Mapping Module Instance".

Figure 8-102 Custom LDAP module instance definition

Click **Finish**. Ignore the Restart WebSphere message at this time.

Replicate the two steps above to define an instance of the PassTicketSTSTModule.

Upon completion, the module instance list should contain the two new instances, as shown in Figure 8-103.

<input type="checkbox"/>	Default WS-Federation Token	SAML11STSTModule
<input type="checkbox"/>	UserName Module Instance	UsernameTokenSTSTModule
<input type="checkbox"/>	AccountJCA-af4c5bfc-010c-eb6c-ed99-8ac836f96d18	AuthorizationSTSTModule
<input type="checkbox"/>	AccountJCA-af4c5e40-010c-f662-2754-8ac836f96d18	SAML20STSTModule
<input type="checkbox"/>	LDAP Mapping Token	LDAPSTSTModule
<input type="checkbox"/>	PassTicket Token	PassTicketSTSTModule
Page 1 of 1		Total: 15 Filtered: 15 Displayed: 15 Selected:

Figure 8-103 Passticket and LDAP module instance created

It is now possible to create a new Federated Identity Manager Security trust service trust chain. Navigate to **Configure Trust Service** → **Trust Service Chain** and click **Create**. Click **Next** to skip the introduction.

As shown in Figure 8-104, provide a chain name and a description that describe the purpose of this new chain.

Chain Identification	
*Chain Name	JCA-JAAS Chain
Description	Trust Chain for JCA JASS Login module for RACF USERID

Figure 8-104 New trust chain definition

On the Chain Lookup Properties panel, fill in the **AppliesTo address** with ITSOBanker2006 and **Issuer address** with fimprincipal, as shown in Figure 8-105.

The screenshot shows the 'Chain Lookup' configuration panel. It is divided into several sections:

- Request Type:** A dropdown menu is set to 'Validate', and the 'Request Type URI' field contains 'http://schemas.xmlsoap.org/ws/'.
- Lookup Type:** Two radio buttons are present. The first, 'Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenTy', is selected. The second is 'Use XPath to Define Custom Lookup Rule'.
- AppliesTo:** This section contains three input fields: 'Address' (filled with 'ITSOBanker2006'), 'Service Name', and 'Port Type'. The 'Service Name' and 'Port Type' fields are followed by empty boxes for additional configuration.
- Issuer:** This section also contains three input fields: 'Address' (filled with 'fimprincipal'), 'Service Name', and 'Port Type'. Similar to the 'AppliesTo' section, the 'Service Name' and 'Port Type' fields have empty boxes next to them.
- Token Type:** A single empty input field for the 'Token Type'.

Figure 8-105 AppliesTo and Issuer definition for the new chain

Click **Next** to proceed to the Chain Assembly window. This is where the sequence of modules is described. For this trust chain, three module instances need to be added in the following order:

1. User name module instance in *validate* mode.
2. LDAP module instance in *map* mode.
3. Passticket module instance in *issue* mode.

For each module instance, select it in the **Module Instance** drop-down menu, select the appropriate mode from the **Mode** drop-down menu and click **Add Selected Module Instance to Chain**.

Figure 8-106 and Figure 8-107 on page 286 show how the chain should look when completed.

Chain Assembly
Construct a chain by selecting an instance and a mode and use the required instances and then click Next.

Module Instance
PassTicket Token

Mode
issue

Add Selected Module Instance to Chain

Trust Service Chain

Delete Reorder

Select	Order	Module Instance Name
<input type="radio"/>	1	UserName Module Instance
<input type="radio"/>	2	LDAP Mapping Token
<input type="radio"/>	3	PassTicket Token
		Total: 3 Selected: 0

Figure 8-106 Token Module instance order in the new chain

Module Instance Name	Module Instance Type	Mode
UserName Module Instance	UsernameTokenSTSTModule	validate
LDAP Mapping Token	LDAPSTSTModule	map
PassTicket Token	PassTicketSTSTModule	issue

Figure 8-107 Token Module instance mode

Click **Next** to configure each module instance following the order in the chain.

In the UserName Token Module Configuration panel, select the check box for **Skip password validation** (Figure 8-108). The user name in the incoming user name token is trusted.

Username Token Module Configuration

Enter the required values to configure the Username Token Module

Skip password validation

Use JAAS for authentication

*JAAS Login Module Alias

WSLogin

*Amount of time the token is valid after being issued (seconds)

300

Figure 8-108 User name token module configuration

Click **Next** to configure the LDAP Attribute Mapping Module. In the ITSOBank environment the RACF user ID will be stored in the uniqueidentifier attribute of the inetorgperson, as shown in Figure 8-109 on page 287.

LDAP Attribute Mapping Module	
Maps a given user id to an LDAP attribute	
*LDAP Host	local.demo.com
*LDAP Port	389
*LDAP Bind User	cn=root
*LDAP Bind Password	••••••••
*LDAP Base DN	c=us
*LDAP Search Attribute	cn
*LDAP Mapping Attribute	uniqueidentifier

Figure 8-109 LDAP attribute mapping module configuration

In the RACF Passticket Module configuration window, select the **Include once in token** and **Include token creation time in token** check boxes.

The Application name used for Passticket generation and validation field must contain the z/OS application ID while the Passticket key must contain the 16 hexadecimal digits representing the shared secret that the z/OS security administrator should provide.

The assertion does not need to be signed, so uncheck the **Enable the signing of the RACF Passticket token** check box (see Figure 8-110).

Module Chain Item Properties

RACF PassTicket Module Configuration

Enter the required values to configure the RACF PassTicket module.

Include nonce in token

Include token creation time in token

*The application name used for PassTicket generation and validation

OMVSAPPL

PassTicket key (16 hexadecimal digits, not used on z/OS machines)

.....

Enable the signing of RACF PassTicket tokens

Select Key for Signing Tokens

.....

OK Cancel

Figure 8-110 RACF passticket module configuration

Click **Next** to review the trust chain summary before completion. Click **Finish** and then restart WebSphere Application Server when prompted.

The All Federated Identity Manager configuration required for the example is complete, and it is time to run the example.

8.5 Running the scenario

In this section, we describe how to run the ITSO Banking Web service client application to retrieve an account balance using the ITSO Banking application Web service.

Two user cases are described, one where the user has permission to invoke the service, and one where the user does not have permission.

For this purpose, we create two users in Access Manager:

- ▶ itsoman, who is granted permission to invoke the Web service due to membership in the ITSUsers group.
- ▶ fabric, who is not a member of the ITSUsers group and is therefore unable to access the service.

The commands in Example 8-8 are used to create these two users in Access Manager.

Example 8-8 Creating users in Access Manager

```
pdadmin sec_master> user create itsoman cn=itsoman,c=us itsoman  
passwOrd ITS0Users  
pdadmin sec_master> user modify itsoman account-valid yes  
pdadmin sec_master> user create fabric cn=fabric,c=us fabric fabric  
passwOrd  
pdadmin sec_master> user modify fabric account-valid yes
```

User itsoman also needs to have an attribute uniqueidentifier that contains his RACF ID. This attribute can be added to his LDAP inetorgperson object by creating a file named itsoman.add, as shown in Example 8-9.

Example 8-9 itsoman.add - LDAP modification script for itsoman

```
dn: cn=itsoman,c=us  
changetype: modify  
add: uniqueidentifier  
uniqueidentifier: ITSOMAN1
```

The LDAP update can be applied with the `idsldapmodify` utility, as shown in Example 8-10.

Example 8-10 Updating itsoman's LDAP entry

```
# idsldapmodify -h local.demo.com -D cn=root -w passwOrd -i itsoman.add  
  
modifying entry cn=itsoman,c=us
```

In any subdirectory on the machine, copy the client application (ITS0BankerClient2006.ear) and the client key store (client.jks).

In the same shell, source the WebSphere Application Server setupCmdLine script and then launch the client application, as shown in Example 8-11.

Example 8-11 Launching the application client

```
# . /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin/setupCmdLine.sh  
# /opt/IBM/WebSphere/AppServer/bin/launchClient.sh  
ITS0BankerClient2006.ear
```

At this point, the client application should appear, as shown in Figure 8-111.

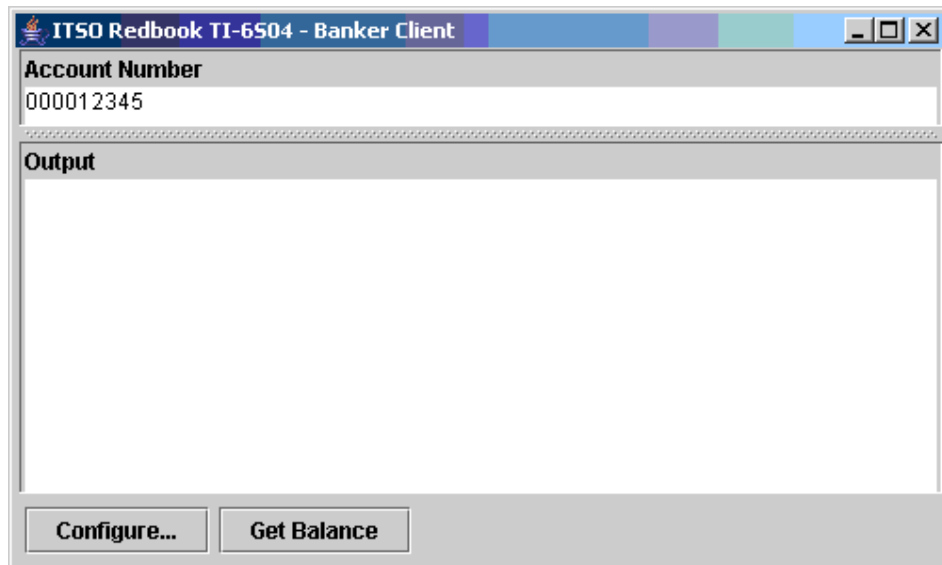


Figure 8-111 Application startup

Click **Configure...** and set the endpoint for the Web service, as shown in Figure 8-112. In the ITSOBank environment, the endpoint is at:

`http://local.demo.com:9080/routerProject/services/AccountJCA`

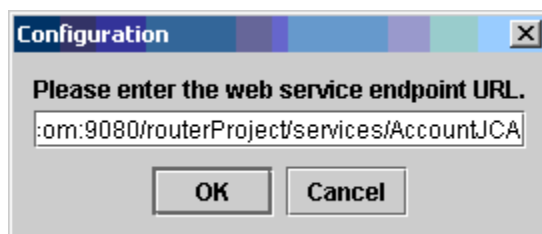


Figure 8-112 Endpoint configuration

Click **OK** to save this configuration. Next, click the **Get Balance** button. When prompted, supply the user name `fabric` and password `passw0rd`, as shown in Figure 8-113 on page 291, and click **OK**.



Figure 8-113 user name and password on Banker Client application

Since fabric is not entitled to invoke the ITSO Banking Web service, a Web service invocation error occurs. Figure 8-114 shows the important details in the resulting error message.

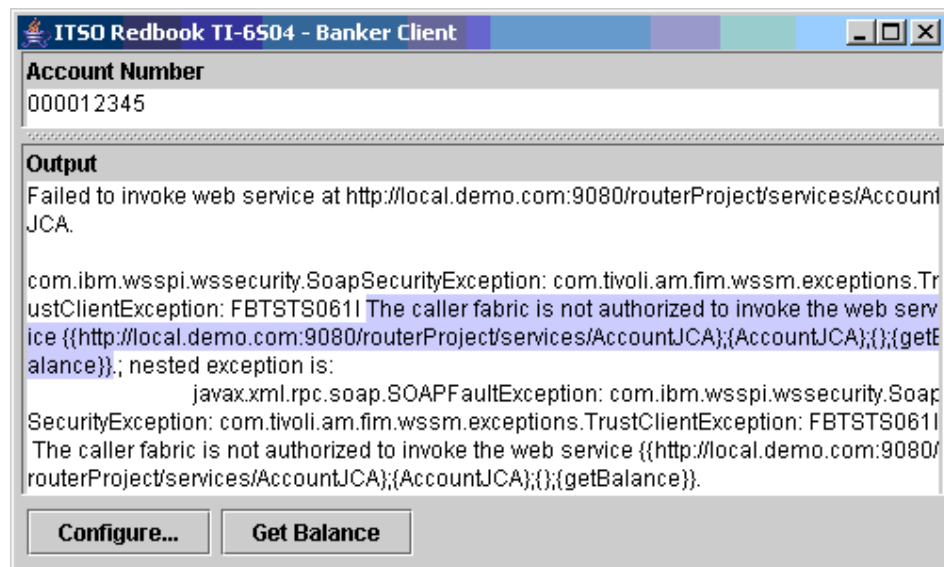


Figure 8-114 Web service invocation failure

Click **Get Balance**. When prompted, supply the user name `itsoman` and password `passw0rd`, as shown in Figure 8-115, and click **OK**.

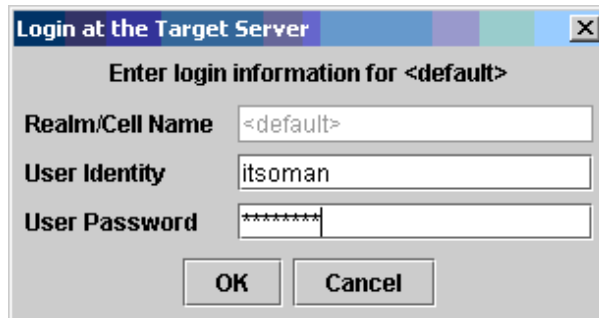


Figure 8-115 Banker Client login for an entitled user

Since `itsoman` has permission to invoke the service, the ITSO Banking Service accepts the request and in turns invokes the execution of a CICS transaction (for more detail on the sequence of steps occurring, refer back to 8.1, “Implementation scope” on page 194).

The CICS transaction runs under the credential of the RACF ID `ITSOMAN1`, and authentication is achieved with the `passticket` provided by the Federated Identity Manager trust chain for the JCA connector.

The transaction results are reported on the client, as shown in Figure 8-116 on page 293.

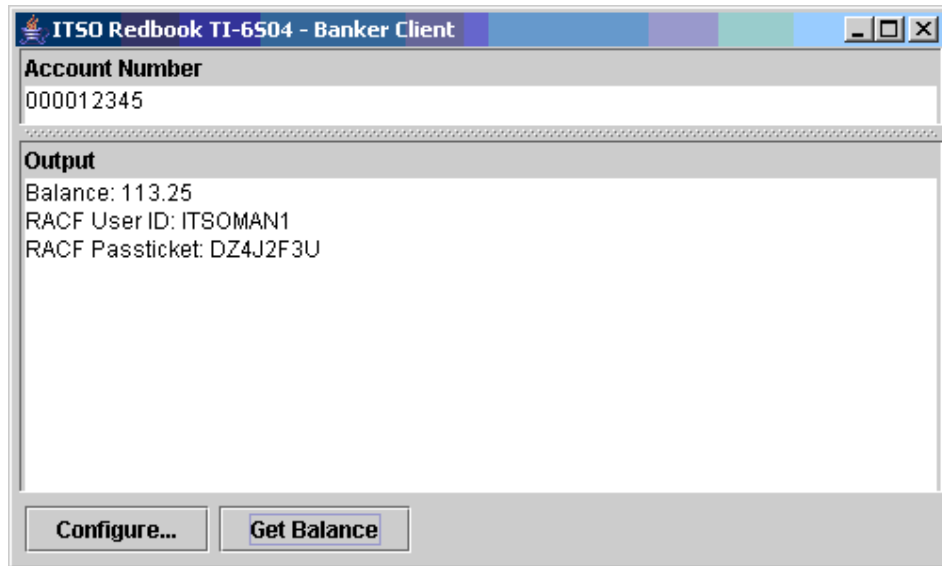


Figure 8-116 Balance results obtained from the CICS transaction server

Content of SOAP messages

In this section, we are providing the SOAP messages that were captured using the TCPMon utility provided with WebSphere Application Server.

Example 8-12 Trace 1

```
<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-wss
ecurity-secext-1.0.xsd">
      <wsse:BinarySecurityToken
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-s
oap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509
-token-profile-1.0#X509" wsu:Id="x509bst_26"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">MIIB2DCCAUGgAwIBAgIERLfvGdANBgkqhkiG9w0BAQQFADA
xMQswCQYDVQQGEwJVUzENMA5GA1UEChMEaXRzbzETMBEGA1UEAxMKYmFua2NsaWVudDAeFw
OwNjA3MTQxOTI0NDhaFw0wOTA0MDkxOTI0NDhaMDEwCzAJBgNVBAYTA1VTMQQwCwYDVQQKE
wRpdHNvMRMwEQYDVQQDEwpiYW5rY2xpZW50IGeMAOGCSqGSIb3DQEBAQUAA4GMADCBiAKB
gGn//+sSH2+xRYeHLfQhKsweJ0LuXyFoQ20MWDpZObwSyKABGuIsiaCZFPSVCEzRq1n17K9
tYOMK/EPNJx+PoRDdAizQ1404EVU7wt6G9c2140ASuAyRpZi4oa3WpRIe0z8EzdFxaScLbR
9hVmrDzVGPptG1Jd5Hkr5XXwUyF8SxAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAH2ZzK3S1k
QsjU/TGBoOC1q5VNFDPgLTz9V4BpaHMs2W2MHJW321PurQ/zNa9Gw6Xrg5RnYfmx2ub1FJ
wC3enf73bcQCbCDqAALw7mbQssU1jPnA8GveK/QDXXC5J8cM6qMQoPo9u5jW1LqybLZr8g
nbu45cf/SrTVBV/KoOYk=</wsse:BinarySecurityToken>
      <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmlsig#">
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509
-token-profile-1.0#X509SubjectKeyIdentifier">YxCw5uQd9UQsEEhysWMEiit8Y4
I=</wsse:KeyIdentifier>
            </wsse:SecurityTokenReference>
          </ds:KeyInfo>
        <CipherData>

<CipherValue>giFPhTK0pePP881mqJnHU8tRMTApFooAwG9Vg3zt0Q32HIwGnAczrSSz1e
wbpiQOxZ0nhY+uu91dm1hCqXMKEtULQtENENAtV4PEFnHAUkcE/N8GA0hNKd1W6BcFhJBwU
jxzE1rRSCI3N00B0IZ/cC81xwyW51aMBczBW61ObQk=</CipherValue>
        </CipherData>
        <ReferenceList>
```

```

        <DataReference URI="#wssecurity_encryption_id_28"/>
        <DataReference URI="#wssecurity_encryption_id_29"/>
    </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="wsse ds xsi
soapenc xsd soapenv "
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_25">
                <ds:Transforms>
                    <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces PrefixList="p728 xsi
soapenc xsd wsu soapenv "
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
                    </ds:Transform>
                </ds:Transforms>
            </ds:Reference>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>pGcTdPhxyDjou1AumrRcwgVUHTc=</ds:DigestValue>
            </ds:Reference>
            <ds:Reference URI="#wssecurity_signature_id_27">
                <ds:Transforms>
                    <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces PrefixList="wsse xsi
soapenc xsd wsu soapenv "
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
                    </ds:Transform>
                </ds:Transforms>
            </ds:Reference>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>ggcaXC3P6q7adL26KyeHk00xeto=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>

    <ds:SignatureValue>R1rS1X9360eFzEpEHFSgX/wUFEY1jCx2n7c1yiw0qnytIvjDXhLD

```



```

L2dq0ZNIInDG7256EJ/zALuxTEuUq7yKxMG+3irrcpRdnutq2zr/h/fm5re7EBEH51t2uNf3
qFYa88fEY7xV0QhVmJy0GtKg0IWwHtbpONcVcNLE1H4Nf9A=</ds:SignatureValue>
  <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:Reference URI="#x509bst_26"
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_28"
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>e451CSwt5Pa0X2RUysvoGLvdIIehhJUu9Lq1yLtV11R1npRdJB4YN1IWeT
zRoQD1Wn7miYMBa317bMOBwdw10e2LVS9dWoGpJqj1Rjg8BMquhKop1ge3YJkzuIifKpXH8
W6sSUiPGML8C4j/Xu5dtP96ns9pb60JI746r8s2TILC1agrqAg24h/MVf8XT+aDB1TfwLsk
X5k1qwG81wrKJEvR91U7dDqBYX01V3IywtXUe9Sb3kmcwfCKanRHEeQLm2ryCMtR1GesM3/
UXE20t0b51eW6Vfwgf7Tgg3Juiphb5YzxeehFk/PARdPUBMWZAIv+RTihLuFcWY0ef6y5bP
xzuaWjtII4KvX6qUednhA0m/DHaEFKGjH7RC22MPNLdg92EPrxs7HSVdSX4b/YbJSRG5uBc
2Qh1hdQHEaZGuUj+9vYRJNzGGF66EoCYhh6IoD7NGUkQS7maVJyzRG7z8LVd4FvWMeZtasL
RtyM+2o=</CipherValue>
    </CipherData>
  </EncryptedData>
</wsse:Security>
  <InternationalizationContext soapenv:mustUnderstand="0"
xmlns="http://www.ibm.com/webservices/InternationalizationContext">
    <Locales xmlns="">
      <Locale>
        <LanguageCode>en</LanguageCode>
        <CountryCode>AU</CountryCode>
      </Locale>
    </Locales>
    <TimeZoneId xmlns="">America/Chicago</TimeZoneId>
  </InternationalizationContext>
</soapenv:Header>
  <soapenv:Body wsu:Id="wssecurity_signature_id_25"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">
    <EncryptedData Id="wssecurity_encryption_id_29"
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>ZyD8KEQnncJAG6tRtPfTx9q2qWFEMlQq00Bsjr9DnqZhTevfjjL9VxUF+Y
qhyHzA8con+UBDHOG4JdprcU8hPRY8Z1/lgrEgajIrC1yc1jI8W/tT/EygdKBBHYAS5mcb
SwsZnbSW3XUjJfu3apKmal6Ip0ZDgq4a/6R5vAwgqc=</CipherValue>
    </CipherData>
    </EncryptedData>
  </soapenv:Body>
</soapenv:Envelope>

```

Note that the SOAP security token and the SOAP message body are encrypted and signed.

Once the message arrives at the WebSphere Application Server SOAP Handler, the WSSM component contacts the Federated Identity Manager trust service for token exchange using its WS-Trust client. The WS-Trust request is shown in Example 8-13.

Example 8-13 Trace 2

```

<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:RequestType
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">http://schemas.
xmlsoap.org/ws/2005/02/trust/Validate</wst:RequestType>
      <wst:Issuer
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wsa:Address
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">urn:itfim:
wssm:tokenconsumer</wsa:Address>
        </wst:Issuer>
      <wsp:AppliesTo
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">

```

```

<wsa:Address>http://local.demo.com:9080/routerProject/services/AccountJ
CA</wsa:Address>
    <wsa:PortType>AccountJCA</wsa:PortType>
    <itfim:OperationName
xmlns:itfim="urn:ibm:names:ITFIM">getBalance</itfim:OperationName>
    </wsa:EndpointReference>
</wsp:AppliesTo>
<wst:Base
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
    <wsse:UsernameToken wsu:Id="wssecurity_signature_id_27"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wss
ecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">
        <wsse:Username>itsoman</wsse:Username>
        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
token-profile-1.0#PasswordText">passw0rd</wsse:Password>
        </wsse:UsernameToken>
    </wst:Base>
</wst:RequestSecurityToken>
</soapenv:Body>
</soapenv:Envelope>

```

Verify that the message contains the user name token for the user *itsoman* with the password. Also note the values of the *AppliesTo* and *Issuer* elements. They should match what was specified in the configuration of the partner and application trust chain created in the Federated Identity Manager console.

The next trace is the response from the Federated Identity Manager trust service to the WSSM trust client and its content is shown in Example 8-14.

Example 8-14 Trace 3

```

<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://sch
emas.xmlsoap.org/ws/2005/02/trust/RSTR/Validate</wsa:Action>
    </soapenv:Header>

```

```

    <soapenv:Body>
      <wst:RequestSecurityTokenResponse
wsu:Id="uuidb07cf0f8-010c-eeff-0a03-aeba7319f07e"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-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
ID="Assertion-uuidb07cf0fc-010c-f34b-081d-aeba7319f07e"
IssueInstant="2006-07-27T14:56:53Z" Version="2.0"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
              <saml:Issuer
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">http://mycomp
any.com</saml:Issuer>
                <saml:Subject>
                  <saml:NameID>itsoman</saml:NameID>
                </saml:Subject>
                <saml:Conditions NotBefore="2006-07-27T14:55:53Z"
NotOnOrAfter="2006-07-27T14:57:53Z">
                  <saml:AudienceRestriction>

<saml:Audience>http://local.demo.com:9080/routerProject/services/AccountJCA</saml:Audience>
                    </saml:AudienceRestriction>
                  </saml:Conditions>
                  <saml:AuthnStatement
AuthnInstant="2006-07-27T14:56:53Z">
                    <saml:AuthnContext>

<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Passw
ord</saml:AuthnContextClassRef>
                      </saml:AuthnContext>
                    </saml:AuthnStatement>
                    <saml:AttributeStatement>
                      <saml:Attribute Name="is_racf_user"
NameFormat="wssm_example">
                        <saml:AttributeValue xsi:type="xs:string"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">TRUE</saml:Attrib
uteValue>
                      </saml:Attribute>
                    </saml:AttributeStatement>
                  </saml:AuthnContext>
                </saml:AuthnStatement>
              </saml:Assertion>
            </wst:RequestedSecurityToken>
          </wst:RequestSecurityTokenResponse>
        </soapenv:Body>

```

```

        <saml:Attribute Name="company_name"
NameFormat="wssm_example">
        <saml:AttributeValue xsi:type="xs:string"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">ITS0</saml:Attrib
uteValue>
        </saml:Attribute>
        </saml:AttributeStatement>
    </saml:Assertion>
</wst:RequestedSecurityToken>
    <wst:RequestedAttachedReference
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-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:2.0:assertion"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">Assertion-uuidb07cf0fc-010c-f34b-081d-aeba7319f0
7e</wss:KeyIdentifier>
            </wss:SecurityTokenReference>
        </wst:RequestedAttachedReference>
    </wst:RequestSecurityTokenResponse>
</soapenv:Body>
</soapenv:Envelope>

```

Note that the response contains a SAML 2.0 assertion. This is a result of the configuration in “Create a WSSM Web service application trust chain” on page 266.

This trace also shows the two attributes `company_name` and `is_racf_user` within the SAML attribute statement. This was defined in the mapping rule in “Configuring the WSSM partner chain” on page 259.

Once the WebSphere Application Server SOAP handler has processed the message, then the ITSO Bank Application EJB is invoked and prepares a request for the CICS Transaction Gateway using the JCA connector. The JCA connector makes use of the JAAS login module containing a WS-Trust client to invoke a second Federated Identity Manager trust chain. That request is shown in Example 8-15.

Example 8-15 Trace 4

```

<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:RequestType
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">http://schemas.
xmlsoap.org/ws/2005/02/trust/Validate</wst:RequestType>
      <wst:Issuer
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wsa:Address
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">fimprincip
al</wsa:Address>
        </wst:Issuer>
        <wsp:AppliesTo
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <wsa:EndpointReference
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
            <wsa:Address>ITS0Banker2006</wsa:Address>
          </wsa:EndpointReference>
        </wsp:AppliesTo>
      <wst:Base
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wss:UsernameToken
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
          <wss:Username>itsoman</wss:Username>
          <wsu:Created
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">2006-07-27T14:56:54Z</wsu:Created>
        </wss:UsernameToken>
      </wst:Base>
    </wst:RequestSecurityToken>
  </soapenv:Body>
</soapenv:Envelope>

```

```
</wst:RequestSecurityToken>
</soapenv:Body>
</soapenv:Envelope>
```

In this SOAP message, you can see the WS-Trust elements `appliedto` and `issuer`. Their values should match those configured in 8.3.3, “Configure a JAAS login module” on page 253.

The trust chain exchanges the user name token containing the Access Manager user ID for a user name token containing the RACF user ID and a generated `passticket`. The response from the trust service is shown in Example 8-16.

Example 8-16 Trace 5

```
<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://sch
emas.xmlsoap.org/ws/2005/02/trust/RSTR/Validate</wsa:Action>
  </soapenv:Header>
  <soapenv:Body>
    <wst:RequestSecurityTokenResponse
wsu:Id="uuidb07cf439-010c-e13f-6012-aeba7319f07e"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">
      <wst:RequestedSecurityToken>
        <wss:UsernameToken
wsu:Id="usernameb07cf438-010c-e02c-b60f-aeba7319f07e"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">
          <wss:Username>ITSOMAN1</wss:Username>
          <wss:Nonce
EncodingType="Base64Binary">t7x05qchEuRrcfvwEQmrPw==</wss:Nonce>
          <wsu:Created>2006-07-27T14:56:54Z</wsu:Created>
          <wss:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurit
y-secext-1.0.xsd#PasswordText">DZ4J2F3U</wss:Password>
        </wss:UsernameToken>
```

```

        </wst:RequestedSecurityToken>
        <wst:RequestedAttachedReference
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
            <wss:SecurityTokenReference>
                <wss:Reference
URI="#usernameb07cf438-010c-e02c-b60f-aeba7319f07e"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-user
name-token-profile-1.0#UsernameToken"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd"/>
            </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>

```

The resulting user name token contains the user ID ITSOMAN1 that matches the LDAP attribute uniqueidentifier of itsoman. The password is DZ4J2F3U, which is a passticket generated by the Federated Identity Manager Passticket module.

Trace 6 in Example 8-17 is the SOAP response message of the ITSO Banking application, after the CICS transaction has completed successfully.

Example 8-17 Trace 6

```

<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-wss
ecurity-secext-1.0.xsd">
            <wsse:BinarySecurityToken
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-s
oap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509
-token-profile-1.0#X509" wsu:Id="x509bst_6"

```



```

xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">MIIByzCCATSgAwIBAgIERLfwHDANBgkqhkiG9w0BAQQFADA
qMQswCQYDVQQGEwJVUzENMAAGA1UEChMEaXRzbzEMMAoGA1UEAxMDd2FzMB4XDTA2MDCxND
E5MjcyNFoXDTA5MDQwOTE5MjcyNFowKjELMAkGA1UEBhMCVVMxDTALBgNVBAoTBG10c28xD
AKBgNVBAMTA3dhczCBnzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEA3sEN8l utdfFnU/t0
iVM4l1wWIEjUPohy1YUYX+ES9MKs9MAJJYrvzREOHln0fVDLXDSzf10GxFqdHcByHLb4ro1
JM9ekMfw+2oqwIoFYg0jNRkeRSd3yv4GbPzHd0x8k/52492QcrRs0x+8U7YorqK1apW/Lg
huKM19S91BLYECAwEAATANBgkqhkiG9w0BAQQFAAOBgQC5YF/OvtUvZg3CMS0Tcwe1iqAfh
30dReJQxqush6Q3+j6iLYqz18mjNOedHumB7TP9jwBj4dwJoXPhM8Z11sFquwkms2ZUW6m7
n2RwrwuzU37271KfNNU9dbv0mSFA0Yw1SikdG8hT3sAn408p0JcXipIxZVSS1SN0BwCCF89D
Oog==</wsse:BinarySecurityToken>
  <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <wsse:SecurityTokenReference>
        <wsse:KeyIdentifier
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509
-token-profile-1.0#X509SubjectKeyIdentifier">QiYzsqG1l8rbR+cc5AdsJ/tJbX
k=</wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    <CipherData>
      <CipherValue>VfWp676IhhKo0da9zsbIom8F8HVOKC7feTGwzmanAbrrJU1G115T2fEwU2
zxKZhf6u+i42rNMsKhAT70/TvuTYSqMPOMGhm+0JKFCPUv+0MJnUSfa1YZO7E7bZh6st88mo
dSW/fyd+KIQOYqZT110LV3vURLI/oPvM+EGNOFX5PM=</CipherValue>
    </CipherData>
    <ReferenceList>
      <DataReference URI="#wssecurity_encryption_id_7"/>
    </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="wsse ds xsi
soapenc xsd soapenv "
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_5">
        <ds:Transforms>

```

```

        <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
        <ec:InclusiveNamespaces PrefixList="p728 xsi
soapenc xsd 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/xmlsig#sha1" />
<ds:DigestValue>abEzTDRHTSwa5flfSQ0BSrbjpZA=</ds:DigestValue>
    </ds:Reference>
</ds:SignedInfo>

<ds:SignatureValue>MVfT/1kemJHh/YkBLbD0WgOdfAdMRxFc7zzaG+d1THDPVSqGTAr5
wgn76uaN4gexIBR4vegWefD0up2CXhNHZ8s5rZ+vg9467JEhRvLIphnwX2gZHyTrWdPGMyB
GTAVaPftpZkSbyP+ZyIFK0AhgSq4349Rc8BHR4odX1rojD6k=</ds:SignatureValue>
    <ds:KeyInfo>
        <wsse:SecurityTokenReference>
            <wsse:Reference URI="#x509bst_6"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509
-token-profile-1.0#X509" />
            </wsse:SecurityTokenReference>
        </ds:KeyInfo>
    </ds:Signature>
</wsse:Security>
</soapenv:Header>
    <soapenv:Body wsu:Id="wssecurity_signature_id_5"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-utility-1.0.xsd">
        <EncryptedData Id="wssecurity_encryption_id_7"
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>2WfCM0BB1uAqZnCeURMzK/HuR2SwxaykBsY0w0Li+L1qfB7XNhpGE0JnzV
2bKf8700Tgn+iD7bkcoQSTopyKGBBI8f99BivpigY8Ax1Kg0/SBj5Zsj9Bdw39B3IKz8f8/
Wj7BG8bnL6IxOMisH6BrpxY/4y0cbEIJuMDnrWoiu41ANBTUv4m9JMhj9NwUJUT+rQgPj3
/lisNTAcVAS0hBIDf35BHBd28Axe8CT6ZhVuqMgkBo1fpItkRQf0+iEJ5kBUG3Lo7m0o53k
w1BT4dwmLMcZAS1nA</CipherValue>
            </CipherData>
        </EncryptedData>
    </soapenv:Body>

```

</soapenv:Envelope>

This response is also signed and encrypted.

This concludes our successful configuration and test of the use case described in Figure 8-1 on page 194.

8.6 Common Auditing and Reporting Service configuration

IBM Tivoli Federated Identity Manager V6.1 and IBM Tivoli Access Manager V6.0 support a common audit infrastructure called *Common Auditing and Reporting Service* (CARS).

This service leverages the standard XML-based format structure of an event called Common Base Event (CBE) and the IBM Common Event Infrastructure (CEI) technology to provide a centralized collection point and a consistent life cycle management of auditable events (archive, restore, and purge).

The Common Auditing and Reporting Service consists of the following components:

- ▶ Server: (common to all exploiting products)
- ▶ Client - There are three types of clients:

- Java client

The Java client is used by some components of Tivoli Access Manager (for example, SMS server).

- C client

The C client is used by some components of IBM Tivoli Access Manager (for example, WebSEAL).

- Embedded client

This component is used by IBM Tivoli Federated Identity Manager, it is packaged as a set of JAR files, and it includes the security event factory and emitter.

The Common Auditing and Reporting Service uses CEI for the submission of events. When such events can be denoted as auditable, the CEI server stores these events in an XML data store. When an event is not auditable, the CEI server stores these events in the default event store.

Figure 8-118 show the Common Auditing and Reporting Service architecture.

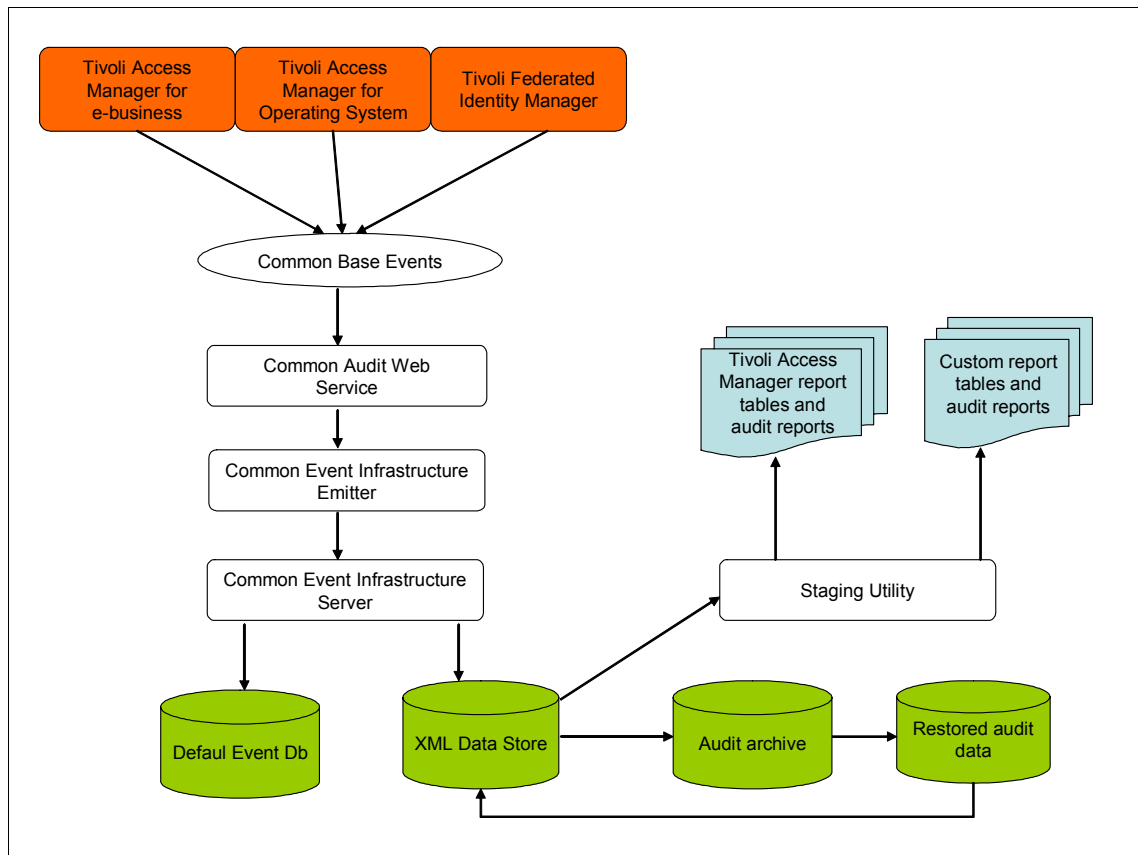


Figure 8-118 The Common Audit and Reporting Service architecture

The Common Auditing and Reporting Service allows staging of data from the XML data store into report tables.

Staging the audit data means parsing the raw data, extracting the interesting pieces, and putting these pieces into staging tables in the database. The data is copied to the staging tables in a format that can be used directly in reports. This process is also called shredding. Records in the report tables can be used by any authoring tool or application to render the audit data in the most appropriate form.

While IBM Tivoli Access Manager for e-business provides some report tables out-of-the-box, custom tables need to be created for IBM Tivoli Federated Identity Manager. Custom tables can be created also for IBM Tivoli Access Manager.

In the next section, we describe how to configure Federated Identity Manager to send audit events to Common Auditing and Reporting Service, how to create custom tables, and how to extract data from those tables to satisfy an audit requirement. The assumption is that the Common Auditing and Reporting Service server is already installed and running as a Web service in WebSphere Application Server.

8.6.1 Configure Federated Identity Manager central auditing

In the ITSOBank environment, the Common Auditing and Reporting Service server is installed on WebSphere Application Server with global security enabled. In this case, a client needs to authenticate using *basic authentication* to WebSphere Application Server in order to access the Common Auditing and Reporting Service Web service.

The Common Auditing and Reporting Service Web service application has a J2EE role defined, hence it is necessary know how users or groups are mapped to this role.

Open the WebSphere Application Server administrative console and navigate to **Applications** → **Enterprise Applications** and select **CommonAuditService**.

Click **Map security roles to users/groups** and verify that at least one user is mapped to the EventSource role. In this scenario, a user named carsman has been created for this purpose (see Figure 8-119).

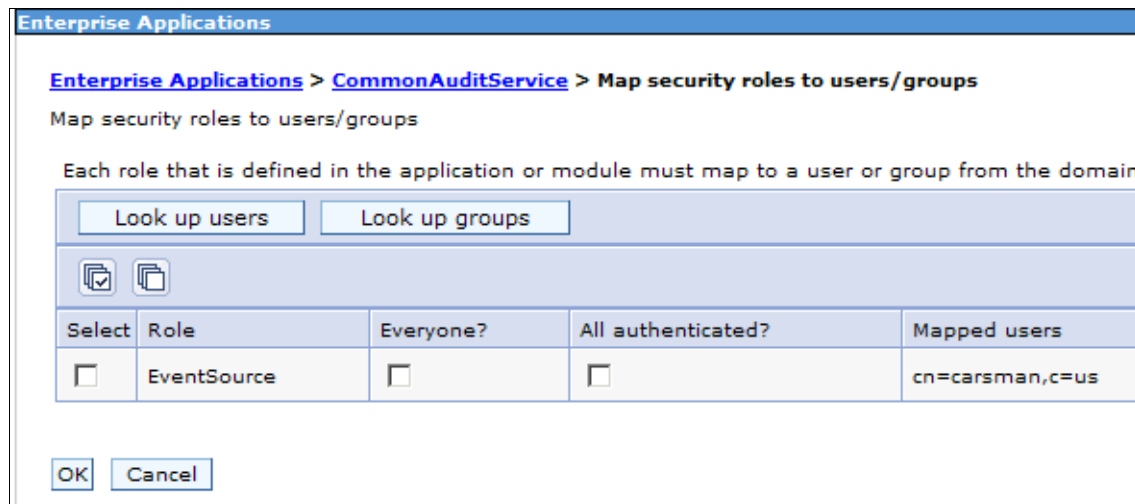


Figure 8-119 Common Auditing and Reporting Service security role mapping

A good verification is to open a browser and point it to the Web service URL, and attempt to authenticate using the identity that is expected to have access. In the ITSOBank environment, the URL is:

`http://local.demo.com:9080/CommonAuditService/services/Emitter`

When prompted, authenticate with user name carsman and the appropriate password. The page should resemble Figure 8-120.

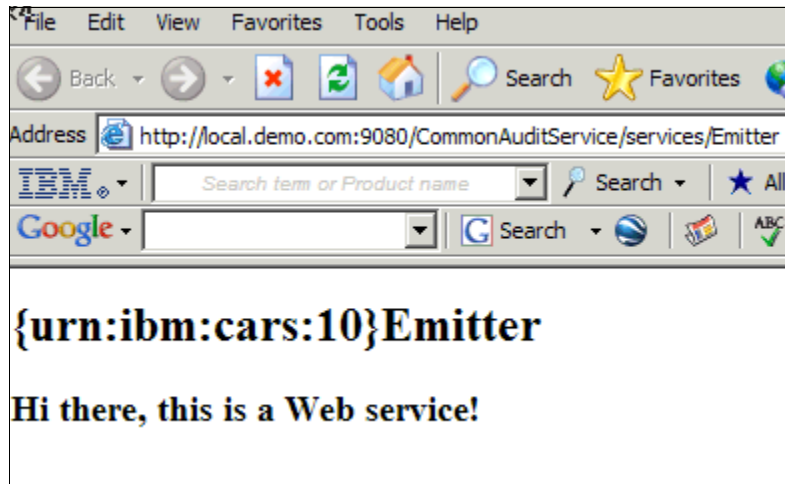


Figure 8-120 Common Auditing and Reporting Service Web service

Once access is verified, open the Federated Identity Manager console and navigate to **Domain Management** → **Auditing**.

In order to enable Federated Identity Manager to send events to the Common Auditing and Reporting Service, select the **Enable audit** check box and select the **Tivoli Common Auditing and Reporting Service** radio button. Provide the complete URL for the Common Auditing and Reporting Service Web service and accept the default for the **Disk cache location**, as depicted in Figure 8-121.

The screenshot shows the 'Auditing' configuration window. On the left, there is a sidebar with 'Audit Settings' and 'Audit Events'. The main area is titled 'Audit Settings' and contains the following configuration options:

- Enable audit**:
- Audit Record Settings**:
 - *Send audit records to:
 - Audit File
 - Tivoli Common Auditing and Reporting Service
- Tivoli Common Auditing and Reporting Service**:
 - *Web Service URL:
 - *Disk cache location:

At the bottom, there is a button labeled 'Web Service Security Settings'.

Figure 8-121 Federated Identity Manager auditing configuration

Click **Web Service Security Settings** and select the **Use Basic Authentication** radio button. Provide the valid user ID and password used in the preceding browser test (see Figure 8-122).

Since HTTP transport is used to access the Common Auditing and Reporting Service Web service, no further information for key store and certificates is required.

The screenshot displays the 'Web Service Security Settings' configuration interface. It is organized into three main sections:

- SSL Settings:** Includes a 'Keystore' dropdown menu set to 'DefaultKeyStore', an empty 'Keystore Password' text field, and a 'List Keys' button.
- Authentication Settings:** Features two radio buttons: 'None' (unselected) and 'Use Basic Authentication' (selected).
- Basic Authentication Settings:** Contains two text input fields: 'Basic Authentication Username' (containing 'carsman') and 'Basic Authentication Password' (masked with dots).

Additional UI elements include a toolbar with icons for adding, deleting, and editing keys, a 'Select Action' dropdown menu, and a table header with columns for 'Select', 'Alias', and 'Key Type'. The table currently shows 'Total: 0', 'Filtered: 0', and 'Display' options.

Figure 8-122 Federated Identity Manager auditing security configuration

Click the **Audit Event** link to show the list of event types that it is possible to audit. Since the usage of the Federated Identity Manager trust service is the primary focus of this scenario, leave this component as the only selected type of event to audit. See Figure 8-123 on page 313 for reference.

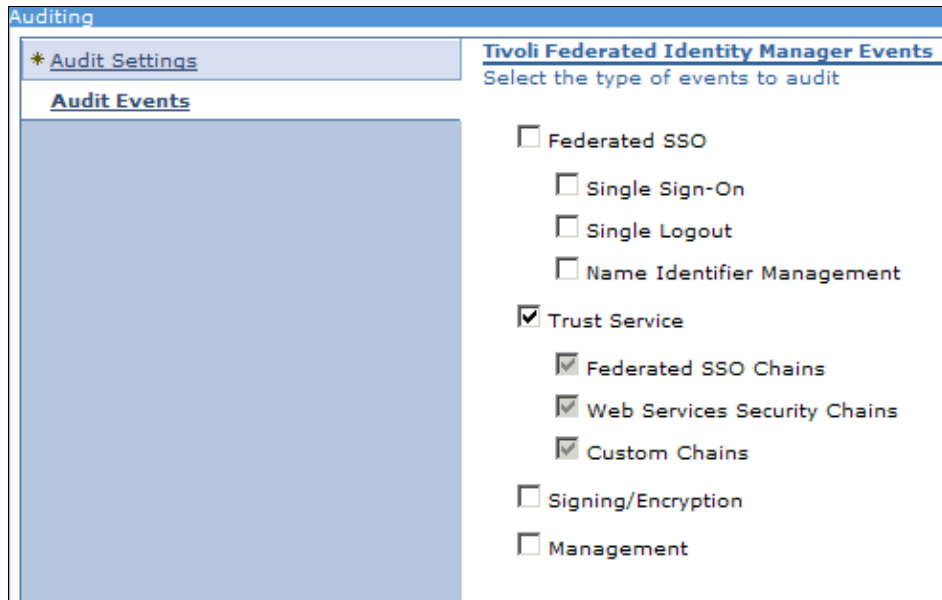


Figure 8-123 Federated Identity Manager audit events

At this point, the audit configuration in Federated Identity Manager has been completed. Click **OK** and restart WebSphere Application Server when advised by the message in the Federated Identity Manager console.

Note: In the ITSOBank environment, the Common Auditing and Reporting Service server and the Federated Identity Manager runtime services are installed on the same WebSphere Application Server instance.

A restart of WebSphere Application Server may cause the Federated Identity Manager runtime service to start before that the Common Auditing and Reporting Service service is available to receive requests. This may result in the following entry in WebSphere Application Server SystemOut.log during the startup:

```
WebServicesFault
  faultCode:
  {http://schemas.xmlsoap.org/soap/envelope/}Server.generalException
  faultString: WSWS3713E: Connection to the remote host
  local.demo.com failed.Received the following error: Connection
  refused
  faultActor: null
  faultDetail:

WSWS3713E: Connection to the remote host local.demo.com
failed.Received the following error: Connection refused
    at
com.ibm.ws.webservices.engine.xmlsoap.builders.WebServicesFaultProce
ssor.createFault(WebServicesFaultProcessor.java:415)
    ....
    ....
    at
com.ibm.cars.events.emitter.soapclient.gen.EmitterStub.sendEvent(Emi
tterStub.java:92)
    ....
```

If this happens it may be possible that Federated Identity Manager will not send events to the Common Auditing and Reporting Service service. In order to work around this problem, stop and start the Federated Identity Manager runtime services application from the WebSphere Application Server administrative console. This might have to be done each time WebSphere Application Server is restarted.

8.6.2 Configuring central auditing for trust service events

It is necessary to create some custom secondary report tables in the XML event store database as a staging area for Federated Identity Manager events. In this scenario, it is enough to create one table for staging trust service events.

Create a Data Definition Language (DDL) file named `customtable.ddl` that defines a new table named `custom_t_trust`. A subset of attributes from the IBM Tivoli Federated Identity Manager `IBM_SECURITY_TRUST` events is staged into the columns of this new `custom_t_trust` table. The contents of the `customtable.ddl` file should match what is shown in Example 8-18.

Example 8-18 customtable.ddl

```
create table custom_t_trust
(
  event_id VARCHAR(64) not NULL,
  cars_seq_number BIGINT,
  appliesTo VARCHAR(1024),
  issuer VARCHAR(1024),
  token VARCHAR(1024),
  moduleName VARCHAR(1024),
  action VARCHAR(1024),
  ruleName VARCHAR(1024),
  tokenInfo VARCHAR(1024),
  accessDecision VARCHAR(1024),
  foreign key (cars_seq_number) references cars_t_event on delete
  cascade
) in cars_ts_16K;
```

Source the `db2profile` for the database instance used for the Common Auditing and Reporting Service database. In this scenario, the same DB2® instance, namely `LDAPDB2`, is used as for the LDAP server instance. Run the DB2 commands to create the new table, as shown in the Example 8-19.

Example 8-19 Creating the new database table.

```
#. /home/ldapdb2/sql1lib/db2profile
# db2 connect to eventxml user ldapdb2 using passw0rd
```

Database Connection Information

```
Database server          = DB2/LINUX 8.2.4
SQL authorization ID    = LDAPDB2
Local database alias    = EVENTXML
```

```
# db2 -tsf customtable.ddl
DB20000I The SQL command completed successfully
```

After the table is created, it is necessary to modify the shredder file so that the staging utility will be able to extract data from the event XML database and stage them into the custom table.

Save a copy of the `$(CARS_HOME)/server/etc/CARSShredder.conf` file that is provided with the installation of the Common Auditing and Reporting Service.

Copy `CARSShredder.conf.custom.template` located in `$(CARS_HOME)/server/template` as `CARSShredder.conf` and place it in `$(CARS_HOME)/server/etc/` directory.

Modify the `$(CARS_HOME)/server/etc/CARSShredder.conf` file to stage additional event attributes necessary for our custom reports. Add the following entries to the `[security_trust]` section, as shown in Example 8-20.

Example 8-20 Additions to CARSShredder.conf.

```
custom_t_trust, event_id, #GLOBAL_ID
custom_t_trust, cars_seq_number, #RECORD_ID
custom_t_trust, appliesTo, CommonBaseEvent/extendedDataElements
[@name='appliesTo']/values
custom_t_trust, issuer, CommonBaseEvent/extendedDataElements
[@name='issuer']/values
custom_t_trust, token, CommonBaseEvent/extendedDataElements
[@name='token']/values
custom_t_trust, moduleName, CommonBaseEvent/extendedDataElements
[@name='moduleName']/values
custom_t_trust, action, CommonBaseEvent/extendedDataElements
[@name='action']/values
custom_t_trust, ruleName, CommonBaseEvent/extendedDataElements
[@name='rule']/values
custom_t_trust, tokenInfo, CommonBaseEvent/extendedDataElements
[@name='tokenInfo']/values
custom_t_trust, accessDecision, CommonBaseEvent/extendedDataElements
[@name='accessDecision']/values
```

The full `[security_trust]` sections should now resemble what is shown in Example 8-21 on page 317.

```
...
;                               EVENT 25. IBM_SECURITY_TRUST
;
[security_trust]
cars_t_event, event_id,          #GLOBAL_ID
cars_t_event, cars_seq_number,  #RECORD_ID
cars_t_event, eventType,       "'AUDIT_TRUST'"
cars_t_event, src_location,
CommonBaseEvent/sourceComponentId/@location
cars_t_event, src_loc_type,
CommonBaseEvent/sourceComponentId/@locationType
cars_t_event, src_comp,
CommonBaseEvent/sourceComponentId/@component
cars_t_event, src_sub_comp,
CommonBaseEvent/sourceComponentId/@subComponent
cars_t_event, src_instance_id,
CommonBaseEvent/sourceComponentId/@instanceId
cars_t_event, app_usr_name,
CommonBaseEvent/extendedDataElements[@name='userInfoList']/children[@name='userInfo']/children[@name='appName']/values
cars_t_event, usr_domain,
CommonBaseEvent/extendedDataElements[@name='userInfoList']/children[@name='userInfo']/children[@name='domain']/values
cars_t_event, usr_loc,
CommonBaseEvent/extendedDataElements[@name='userInfoList']/children[@name='userInfo']/children[@name='location']/values
cars_t_event, usr_loc_type,
CommonBaseEvent/extendedDataElements[@name='userInfoList']/children[@name='userInfo']/children[@name='locationType']/values
cars_t_event, usr_session_id,
CommonBaseEvent/extendedDataElements[@name='userInfoList']/children[@name='userInfo']/children[@name='sessionId']/values
cars_t_event, outcome_result,
CommonBaseEvent/extendedDataElements[@name='outcome']/children[@name='result']/values
cars_t_event, outcome_fail_rsn,
CommonBaseEvent/extendedDataElements[@name='outcome']/children[@name='failureReason']/values
cars_t_event, time_stamp,        CommonBaseEvent/@creationTime
cars_t_event, start_time,
CommonBaseEvent/extendedDataElements[@name='startTime'][@type='dateTime']/values
```

```

cars_t_event, end_time,
CommonBaseEvent/extendedDataElements[@name='endTime'][@type='dateTime']
/values
custom_t_trust, event_id, #GLOBAL_ID
custom_t_trust, cars_seq_number, #RECORD_ID
custom_t_trust, appliesTo, CommonBaseEvent/extendedDataElements
[@name='appliesTo']/values
custom_t_trust, issuer, CommonBaseEvent/extendedDataElements
[@name='issuer']/values
custom_t_trust, token, CommonBaseEvent/extendedDataElements
[@name='token']/values
custom_t_trust, moduleName, CommonBaseEvent/extendedDataElements
[@name='moduleName']/values
custom_t_trust, action, CommonBaseEvent/extendedDataElements
[@name='action']/values
custom_t_trust, ruleName, CommonBaseEvent/extendedDataElements
[@name='rule']/values
custom_t_trust, tokenInfo, CommonBaseEvent/extendedDataElements
[@name='tokenInfo']/values
custom_t_trust, accessDecision, CommonBaseEvent/extendedDataElements
[@name='accessDecision']/values
;
.....

```

At this point, the Common Auditing and Reporting Service server and Federated Identity Manager are ready for auditing trust service events.

Repeat the execution scenario described in 8.5, “Running the scenario” on page 288 to generate some audit events from the Federated Identity Manager trust service. The audit events are stored by the CEI server in the event XML database in a compressed form.

A DB2 stored procedure (IBMCARS_DD_REPORT(id)) that comes with the Common Auditing and Reporting Service can be used to retrieve raw data from the event XML database, but its usefulness is limited without further filtering on the events themselves. Using the staging utility, it is possible to populate the custom database with only the data needed to create a detailed report.

In this scenario, a third-party reporting tool is not used. The use of the audit data is via an SQL statement to retrieve the information needed.

The staging utility is a Java class `com.ibm.cars.staging.Staging`, which can be run in:

- ▶ Historical mode
All events in a specified time range are staged. For this mode, the start and end time must be specified.
- ▶ Incremental mode
All new events since the last incremental staging are staged. If incremental staging has never run, all events in the event XML database are staged to the proper staging tables.
- ▶ Prune mode
All events older than a specified time are deleted, or pruned. For this mode you must specify a time and date and all events created before that date are pruned.

In order to run the staging utility, it is necessary to define a proper class path. Example 8-22 shows a sample script, `stage.sh`, that sets the correct environment and runs the utility.

Example 8-22 stage.sh - Script to run the Common Auditing and Reporting Service staging utility

```
#!/bin/ksh

echo "sourcing WAS setupCmdline.sh "

. /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin/setupCmdLine.sh

echo "sourcing the db2 profile"
. /home/ldapdb2/sql1lib/db2profile

echo "Setting classpath"

CLASSPATH=/opt/IBM/Tivoli/CommonAudit/server/etc:/opt/IBM/Tivoli/Common
Audit/server/lib/ibmcars.jar:/opt/IBM/db2/V8.1/java/db2jcc.jar:/opt/IBM
/db2/V8.1/java/db2jcc_license_cu.jar:/home/ldapdb2/sql1lib/java/db2java.
zip:/home/ldapdb2/sql1lib/java/db2jcc.jar:/home/ldapdb2/sql1lib/function:
/home/ldapdb2/sql1lib/java/db2jcc_license_cu.jar:.
export CLASSPATH

echo "staging now..."
```

```
java com.ibm.cars.staging.Staging -dbpassword $1 -mode incremental  
  
echo "Done."
```

If you have created this script, it is necessary to invoke it providing the password of the database instance owner (ldapdb2 in the ITSOBank scenario).

Running this utility can take some time if there are many events to be processed. A valid output would resemble what is shown in Example 8-23.

Example 8-23 Sample output from the staging script

```
# ./stage.sh passw0rd  
sourcing WAS setupCmdline.sh  
sourcing the db2 profile  
Setting classpath  
staging now...  
CBASU0137I Starting XML shredder initialization.  
CBASU0140I Processing configuration section  
[IBM_CBA_AUDIT_PASSWORD_CHANGE]  
CBASU0140I Processing configuration section [IBM_SECURITY_SIGNING]  
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_AUTHZ]  
CBASU0140I Processing configuration section  
[IBM_SECURITY_AUTHN_CREDS_MODIFY]  
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_COMPLIANCE]  
CBASU0140I Processing configuration section  
[IBM_CBA_AUDIT_AUTHN_TERMINATE]  
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_RUNTIME_KEY]  
CBASU0140I Processing configuration section [IBM_SECURITY_AUTHN]  
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_AUTHN]  
CBASU0140I Processing configuration section  
[IBM_CBA_AUDIT_RESOURCE_ACCESS]  
CBASU0140I Processing configuration section [IBM_SECURITY_ENCRYPTION]  
CBASU0140I Processing configuration section  
[IBM_CBA_AUDIT_AUTHN_CREDS_MODIFY]  
CBASU0140I Processing configuration section [IBM_SECURITY_TRUST]  
CBASU0140I Processing configuration section [IBM_SECURITY_AUTHZ]  
CBASU0140I Processing configuration section [IBM_SECURITY_MGMT_POLICY]  
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_DATA_SYNC]  
CBASU0140I Processing configuration section  
[IBM_CBA_AUDIT_MGMT_PROVISIONING]  
CBASU0140I Processing configuration section  
[IBM_CBA_AUDIT_MGMT_REGISTRY]
```



```
CBASU0140I Processing configuration section
[IBM_CBA_AUDIT_AUTHN_MAPPING]
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_MGMT_POLICY]
CBASU0140I Processing configuration section [IBM_SECURITY_FEDERATION]
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_RUNTIME]
CBASU0140I Processing configuration section
[IBM_CBA_AUDIT_MGMT_RESOURCE]
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_MGMT_CONFIG]
CBASU0140I Processing configuration section
[IBM_SECURITY_AUTHN_TERMINATE]
CBASU0140I Processing configuration section [IBM_SECURITY_RUNTIME]
CBASU0140I Processing configuration section [IBM_CBA_AUDIT_WORKFLOW]
CBASU0138I Starting shredder complete.
CBASU0136I Processing tables cei_t_xml01 / cei_t_xmlx01.
CBASU0111I No events need to be staged or pruned. Exiting.
CBASU0136I Processing tables cei_t_xml00 / cei_t_xmlx00.
CBASU0141I 7 events were staged.
```

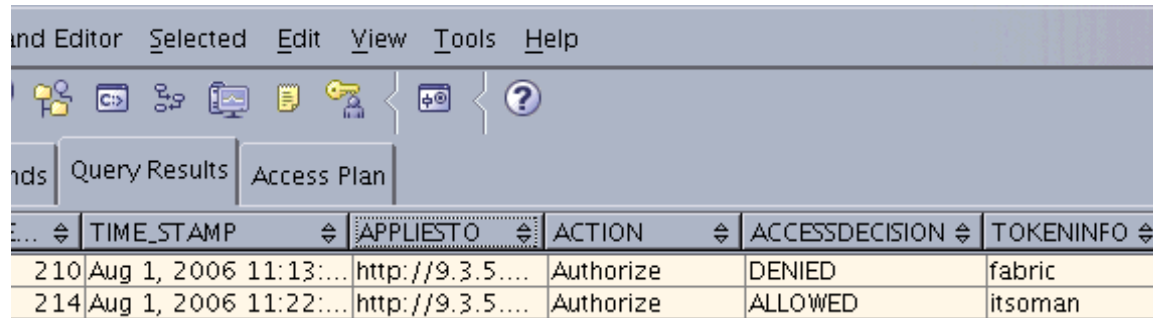
Once the utility has staged some events, it is possible to run an SQL command to retrieve the information.

Suppose that the users who have successfully or unsuccessfully invoked the ITSO Web Banking Service need to be identified. The SQL statement in Example 8-24 has been used in the ITSOBank environment to extract this information.

Example 8-24 Sample SQL statement

```
SELECT t1.cars_seq_number, time_stamp, appliesTo, issuer,
accessdecision, moduleName, action, ruleName, tokenInfo FROM
ldapdb2.cars_t_event t1, ldapdb2.custom_t_trust t2 WHERE
t1.cars_seq_number=t2.cars_seq_number AND t2.action='Authorize' BETWEEN
'2006-08-01-11.10.00.000000' AND '2006-08-01-11.30.00.000000'
```

If this command is run from the DB2 Control Center, Figure 8-124 shows the output.



The screenshot shows a software interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. Below the toolbar are tabs for 'Query Results' and 'Access Plan'. A table displays the results of an audit event query. The table has six columns: ID, TIME_STAMP, APPLIEDTO, ACTION, ACCESSDECISION, and TOKENINFO. Two rows of data are visible.

ID	TIME_STAMP	APPLIEDTO	ACTION	ACCESSDECISION	TOKENINFO
210	Aug 1, 2006 11:13:...	http://9.3.5....	Authorize	DENIED	fabric
214	Aug 1, 2006 11:22:...	http://9.3.5....	Authorize	ALLOWED	itsoman

Figure 8-124 Sample output from audit event query

Using a slightly modified SQL query, trust events for Map or Validate operations could also be displayed.

8.7 Conclusion

This concludes our end-to-end working example representing the direct exposure of existing CICS applications as services and securing the exposed service realization example for the IBM SOA Foundation Service Creation scenario.



Introduction to service-oriented architecture

This appendix introduces service-oriented architecture (SOA) from a business and architecture perspective.

The appendix is organized into the following sections:

- ▶ Service-oriented architecture overview
- ▶ Getting started with SOA
- ▶ Web services and SOA

Service-oriented architecture overview

This section includes an overview for a service-oriented architecture (SOA). First, we define the key terms and components used to describe an SOA. Second, we review the key challenges and drivers for SOA. Third, we highlight the reasons why SOA is the right choice now. Lastly, we describe an example scenario for building a solution using an SOA approach.

Definition of a service-oriented architecture

Figure A-1 highlights the key terms used to describe a service-oriented architecture.

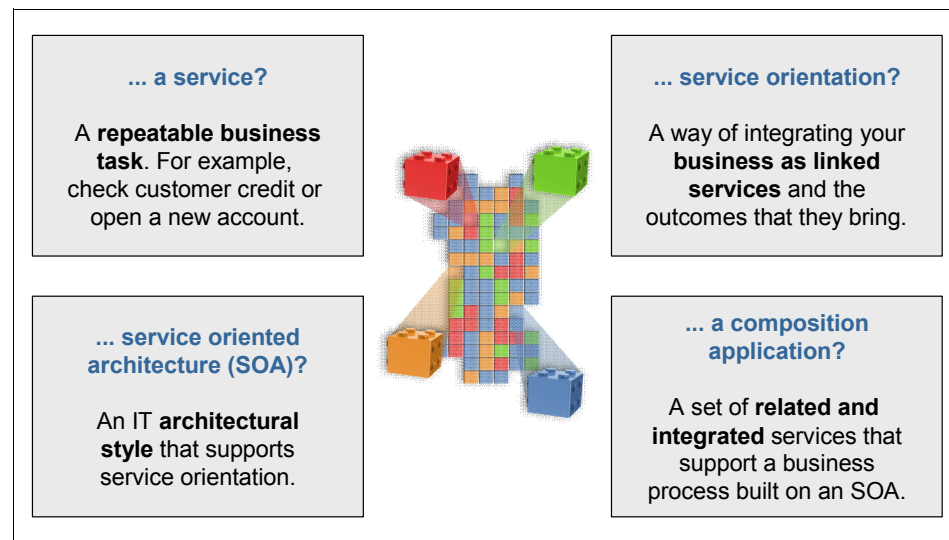


Figure A-1 Definition of key terms for a service-oriented architecture

A *service* is representative of a repeatable business task. Services are used to encapsulate the functional units of an application by providing an interface that is well defined and implementation independent. Services can be invoked (consumed) by other services or client applications.

Service orientation defines a method of integrating business applications and processes as linked services.

Service-oriented architecture (SOA) can be different things to different people depending on the person's role and context (business, architecture, implementation, and operational). From a business perspective, SOA defines a set of business services composed to capture the business design that the enterprise wants to expose internally, as well as its customers and partners.

From an architecture perspective, SOA is an architectural style that supports service orientation. At an implementation level, SOA is fulfilled using a standards based infrastructure, programming model, and technologies, such as Web services. From an operational perspective, SOA includes a set of agreements between service consumers and providers that specify the quality of service, as well as reporting on the key business and IT metrics.

A *composite application* is a set of related and integrated services that support a business process built on an SOA.

Basic components of an SOA

At the most basic level, an SOA consists of the following three components:

- ▶ Service provider
- ▶ Service consumer
- ▶ Service registry

Each component can also act as one of the two other components. For example, if a service provider needs additional information that it can only acquire from another service, it acts as a service consumer. Figure A-2 shows the operations each component can perform.

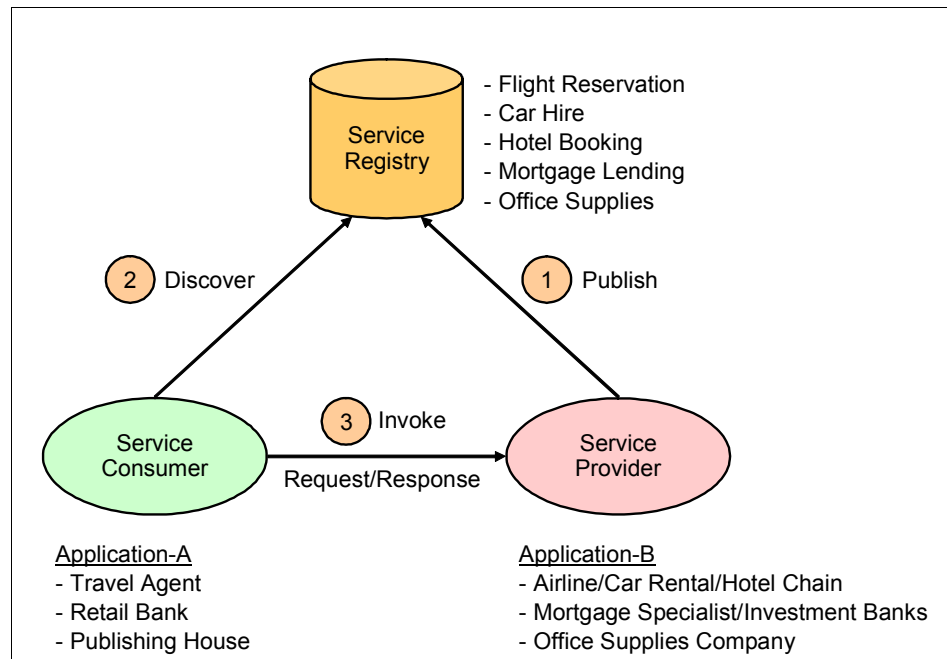


Figure A-2 SOA components and operations

The *service provider* creates a service and in some cases publishes its interface and access information to a service registry.

Each provider must decide which services to expose, evaluate trade-offs between security and easy availability, and determine how to price the services or determine how to exploit the value of the services if they are free. The provider also has to decide in which category the service should be listed, and what sort of trading partner agreements are required to use the service.

The *service registry* is responsible for making the service interface and implementation access information available to service consumers.

The implementers of a service registry must consider the scope with which the registry will be implemented. For example, there are public service registries available over the Internet to an unrestricted audience, as well as private service registries that are only accessible to users within a company-wide intranet.

The *service consumer* locates (discovers) entries in the service registry and then binds to the service provider in order to invoke the defined service.

Challenges and drivers for SOA

In March 2006, IBM commissioned a Global CEO survey and found that 78% of CEOs surveyed believe that integrating business and technology is fundamental for innovation. Another key finding from this survey was that only one in ten CEOs believes his or her organization has the ability to be very responsive to changing market conditions.

As noted from the survey, businesses need the ability to integrate business and technology rapidly to achieve their business objectives. Businesses also have a strong desire to leverage the investment of existing business applications and systems without a complete and costly rewrite. There are many schemes that exist today to integrate systems within and between enterprises. In most cases these solutions are proprietary, not easily adaptable, and not responsive to rapid changes needed by the business.

There is a growing demand for an architecture and technologies that support the connection or sharing of resources and data in a very flexible and industry-standard manner. There is a need to further structure large applications into building blocks that can be reused and composed into business processes.

A shift towards a service-oriented approach standardizes the interaction with applications and business processes, and allows for more flexibility in the process. By adopting an SOA approach, existing application functionality can be turned into reusable services that can be consumed by a new set of client

applications and users. SOA brings the flexibility vital to realizing innovation and desired outcomes of business.

Tip: The alignment of IT with business goals can be summarized as collaborative business and IT decision-making that ensures the following:

- ▶ IT investments are made based on business objectives.
- ▶ IT service delivery provides a business result.
- ▶ Business priorities are assessed with IT capabilities and limitations in mind.

Business requirements and drivers for SOA

Figure A-3 highlights the common elements of a business that require flexible integration.

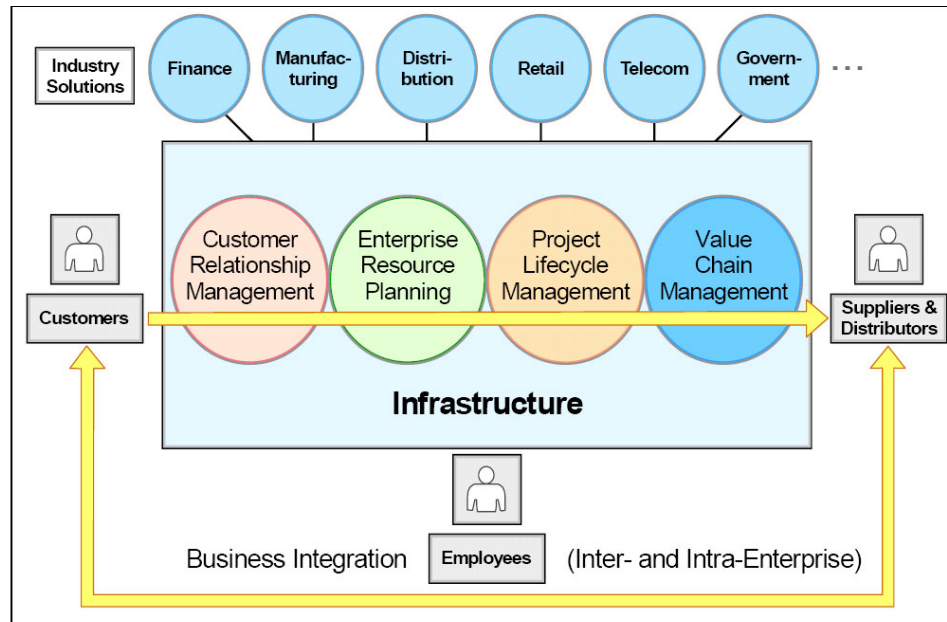


Figure A-3 Business requirements

Here, we summarize the common business drivers that require rapid and flexible integration of IT systems:

- ▶ Support an agile business model.

The marketplace can be very dynamic and competitive. There is a great need to have a business model and IT architecture that can rapidly change to support the business model and its objectives.

- ▶ Reduce cycle time and costs.

Eliminate duplicate systems by reusing existing applications. This has the effect of reducing the time required to integrate systems, reduces cost, and simplifies the skill set required to implement the solution.

When applying these concepts to external business processes, enterprises can move from costly manual transactions to automated transactions with suppliers.

- ▶ Simplify integration across the enterprise.

Many existing IT systems can be inhibitors to change. They are too complex and, as a result, inflexible. Also, existing integration includes multiple technologies and point-to-point integration, which is often inflexible. The need to simplify integration is essential, especially considering the challenges raised from events such as business mergers and acquisitions.

- ▶ Achieve better IT use and return on investment.

Return on investment (ROI) is a comparison of profit earned or lost for the investment with the amount invested. The investment in IT should facilitate the business objective and help the business achieve the targeted ROI.

Greater need for a flexible architecture

There are many possible reasons that a flexible business model is needed, such as business transformation, business process outsourcing, mergers, and acquisitions. SOA provides a flexible IT infrastructure and on demand operating environment to support the initiatives of a flexible business model.

For the purposes of comparison with SOA, we highlight the integration deficiencies of monolithic (silos) and component-based architectures. Next, we describe the flexibility gained by using an SOA approach.

Historically, business applications were built with a monolithic purpose (silos). While this kind of architecture can be effective, it is often very difficult to change and integrate with other applications within the enterprise and between enterprises (custom coded connections required).

For example, a monolithic business application must periodically synchronize inventory information, as you can see in Figure A-4 on page 329. In this approach, pricing information for each Web order is inserted differently based on the application structure. Lastly, there is no common customer or inventory database to be shared across the enterprise, or flexibility in the business processes.

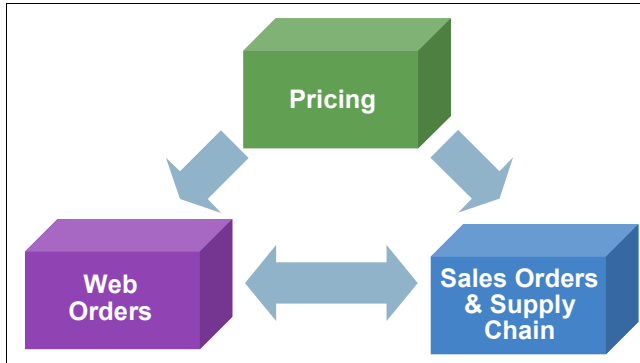


Figure A-4 Monolithic business application (silos)

Although component-based application architecture does define services as units of business logic, there are some inherent problems with this approach. The flow of control is bound into the service logic. The transformation of data formats is also bound to the service logic. There is tight coupling between the services, as seen in Figure A-5, thus making this application integration architecture fragile, giving it a spaghetti-like appearance.

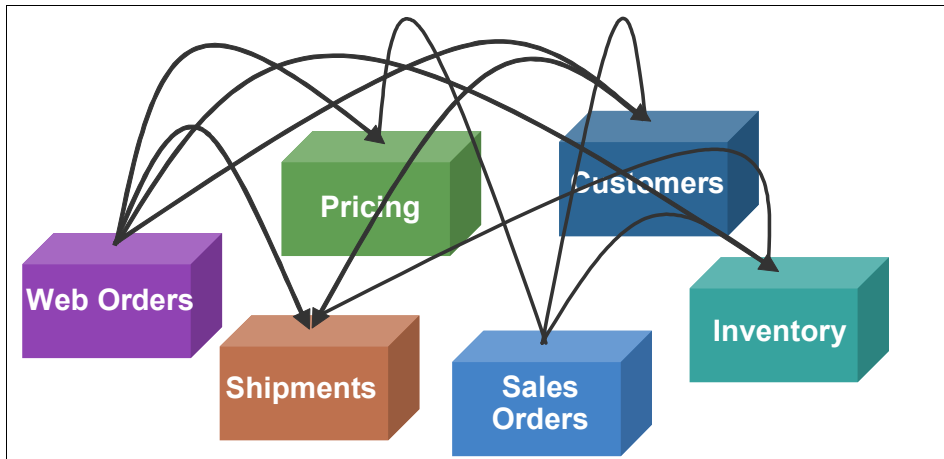


Figure A-5 Component based application

When using an SOA approach (see Figure A-6), the services are defined as units of business logic separated from the flow of control and routing, and the data transformation and protocol transformation. This approach provides loose coupling, thus making this approach much more flexible for integration.

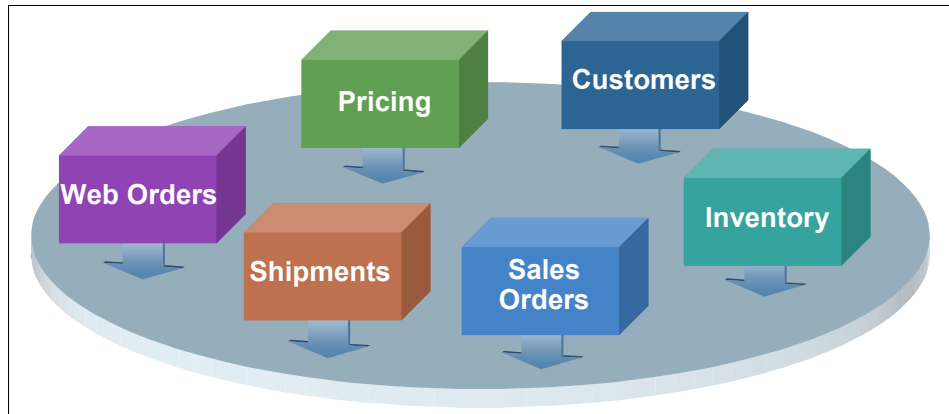


Figure A-6 SOA-based application

Why SOA now

In the previous section, we explained how a service-oriented architecture provides the flexibility to align your IT with your business goals. In this section, we explain why SOA is the right choice now. When there is a shift in architecture, it is important to understand why a shift is needed, and evaluate the maturity level of the architecture that supports adoption.

We highlight the following key reasons why SOA is the right choice now:

- ▶ Business driving a shift in IT
- ▶ Enables flexibility of both IT and business
- ▶ Open standards and platforms
- ▶ Best practices
- ▶ Software for SOA

Business driving a shift in IT

Table A-1 on page 331 provides a summary of business needs that are driving a shift in IT from function-oriented to process- and service-oriented to achieve flexibility.

Table A-1 Shift in IT driven by business

From function-oriented	To process and service-oriented
Build for permanence	Build to change
One long development cycle	Incremental development cycle
Application silos	Orchestrated solutions that work together
Tightly coupled	Loosely coupled
Structure applications using components and objects	Structure applications using services
Known implementation	Implementation abstraction

Enables flexibility of both IT and business

SOA enables flexibility of both IT and business through flexible connectivity of business services:

- ▶ Represent applications or data as a service with a standardized interface.
- ▶ Enable applications as services to exchange structured information (messages, documents, and other business objects).
- ▶ Mediate the message exchange through an Enterprise Service Bus (ESB).
- ▶ Provide on-ramps to the bus for existing applications and systems.

Open standards and platforms

Another key reason that SOA is the right choice for your enterprise, is that it is based on open standards and platforms, as summarized in Figure A-7. These open standards are widely adopted across the industry.

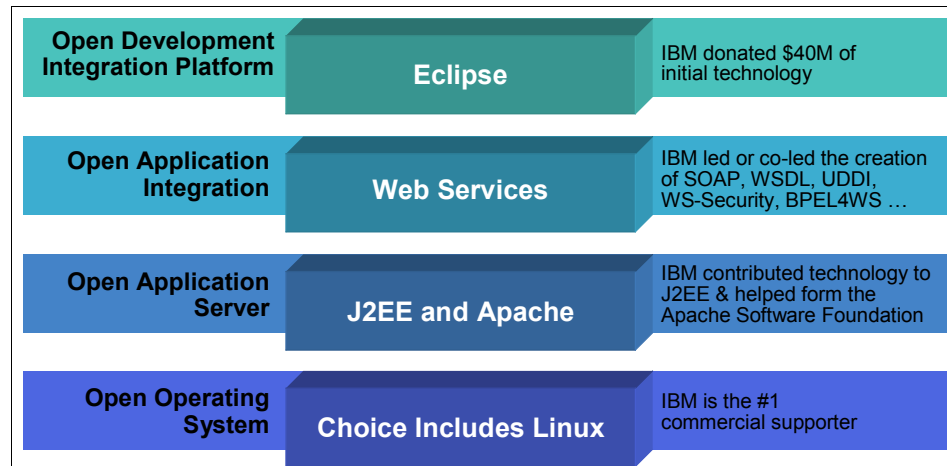


Figure A-7 Summary of SOA open standards and platforms

IBM continues to be a leader in SOA based technologies, products, and solutions. IBM is a key partner in helping define the specifications and technologies used to implement an SOA, such as Web Services, Service Component Architecture (SCA) and Service Data Objects (SDO).

Best practices

Best practices are used to deliver a particular outcome by leveraging the knowledge learned from experience. Best practices include methodologies, techniques, guidelines, and patterns. By leveraging the knowledge captured in best practices listed here, your project can be run with less problems and be deployed more rapidly.

Here is a list of best practices in use today:

- ▶ SOA Adoption
 - The SOA adoption process provides guidelines that assist in developing a roadmap towards a successful migration to an SOA.
- ▶ SOA Governance
 - SOA Governance helps clients extend the planned SOA across the enterprise in a controlled manner.

- ▶ **Methodology**

A well-established set of methodologies can help to break down complex problems into smaller and more manageable pieces that are easier to analyze and, therefore, develop solutions. Example methodologies include the Component Business Model™ (CBM), IBM Service Integration Maturity Model (SIMM), Rational Unified Process® (RUP®), and Service-Oriented Modeling and Architecture (SOMA).
- ▶ **Process modeling**

Process modeling is used to define business processes. A processes flow is a sequence of tasks and decision elements with multiple branches, linked by connectors.
- ▶ **Model-driven development**

Model-driven development is a style of software development where the primary software artifacts are models from which code and other artifacts are generated. A model is a description of a system from a particular perspective, omitting irrelevant detail so that the characteristics of interest are more clear.
- ▶ **Reference architecture**

A reference architecture provides the underlying architecture components used to overcome the initial problems of finding an architecture with which to begin. The most notable reference architecture for SOA is the IBM SOA Foundation.
- ▶ **Patterns**

As a general principle, starting from the beginning each time should be avoided. The use of *patterns* is one specific form of capturing and reusing reoccurring design elements. For example, the Patterns for e-business include reusable architecture and implementation assets used to accelerate the creation of a solution design and implementation.

Software for SOA

The marketplace offers many software choices for SOA. IBM is the market leader in providing mature software and solutions for SOA.

SOA approach for building a solution

This section includes an example SOA approach for building a solution. In this example, the company wants to implement a new business process to support customers who are placing orders from an Internet Web site.

The company has existing retail, warehouse, and billing systems, as seen in Figure A-8. The company would like to build new business processes by reusing the functionality provided by the existing systems rather than having to write new applications or new proprietary interfaces to the existing systems.

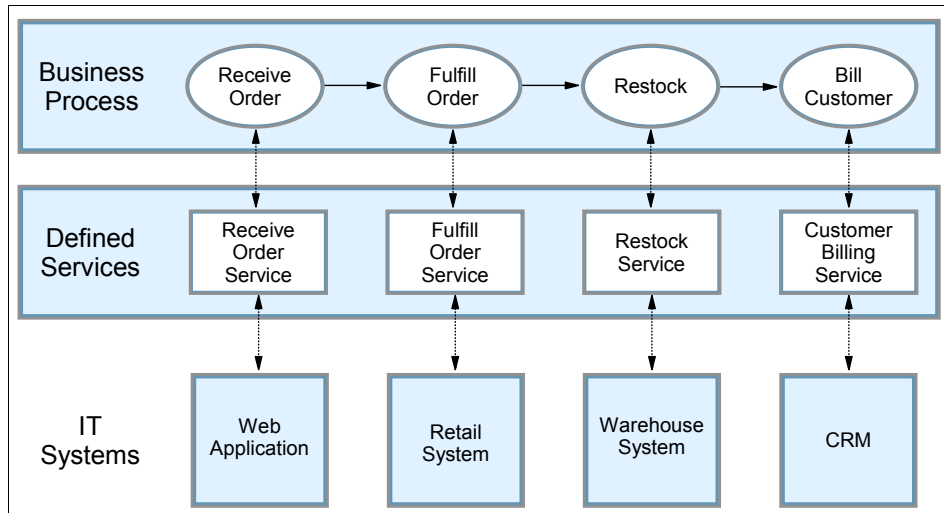


Figure A-8 Service-oriented approach to building systems

If the company has already adopted an SOA approach, it will have defined the interfaces to its existing systems in terms of the functions or services that they offer in support of building business processes. The defined interfaces make building the new system Web front end very simple. The company simply needs to develop an application that invokes (consumes) services to complete the new business process.

By using an SOA approach, companies are able to build horizontal business processes that integrate systems, people, and processes from across the enterprise quickly and easily in response to changing business needs.

Getting started with SOA

In this section, we explore the question of how to get started with SOA from both a business and an architectural perspective.

SOA adoption

SOA adoption provides an iterative and incremental process, and guidelines that assist in developing a road map towards a successful migration to SOA. As seen in Figure A-9, the SOA adoption process begins by defining the scope of possible projects that fit the criteria for being a good fit for a service-oriented architecture.

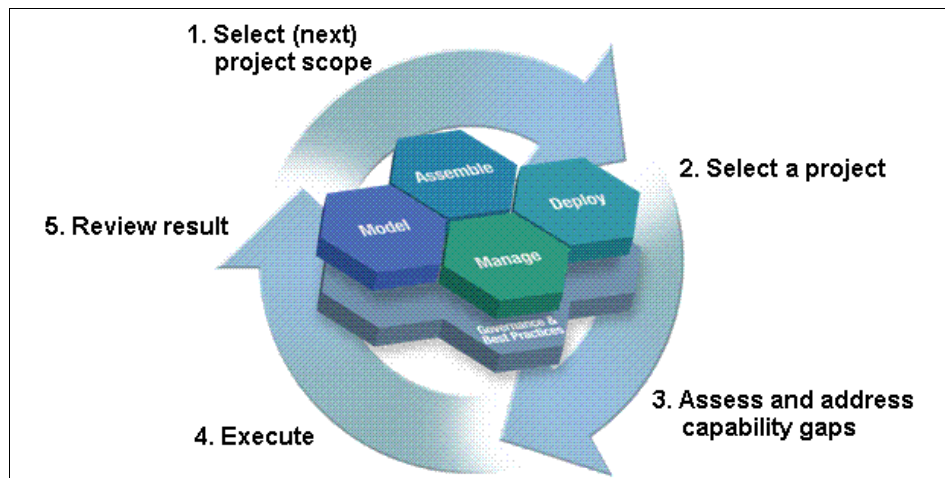


Figure A-9 SOA adoption process

There are two primary perspectives, including strategic vision and project plan. The *strategic vision* perspective describes the business and IT statement of direction, which can be used as a guideline for decision making, organizational buy-in, and standards adoption. The *project plan* perspective (or *tactical* perspective) refers to implementation projects to meet immediate needs of the current business drivers.

Defining the strategic vision starts with assessing the business current maturity across multiple dimensions including business, methodology, and technical. The IBM Service Integration Maturity Model (SIMM) can be used to help in this assessment. If you are more comfortable with starting with a self assessment, you can use the IBM online SOA Assessment Tool:

<http://www.ibm.com/software/solutions/soa/soassessment/index.html>

After the assessment has been performed, the business must establish targets for where they want to be. This includes documenting important goals and metrics for transition across the maturity dimensions. In addition, it is important to have regular checkpoints to reassess the vision.

IBM SOA entry points

As seen in Figure A-10, SOA connects people, processes, and information. To help customers get started with SOA, IBM has defined three core business-centric starting points (people, processes, and information), and two IT-centric starting points (connectivity and reuse). These are known as the SOA entry points.

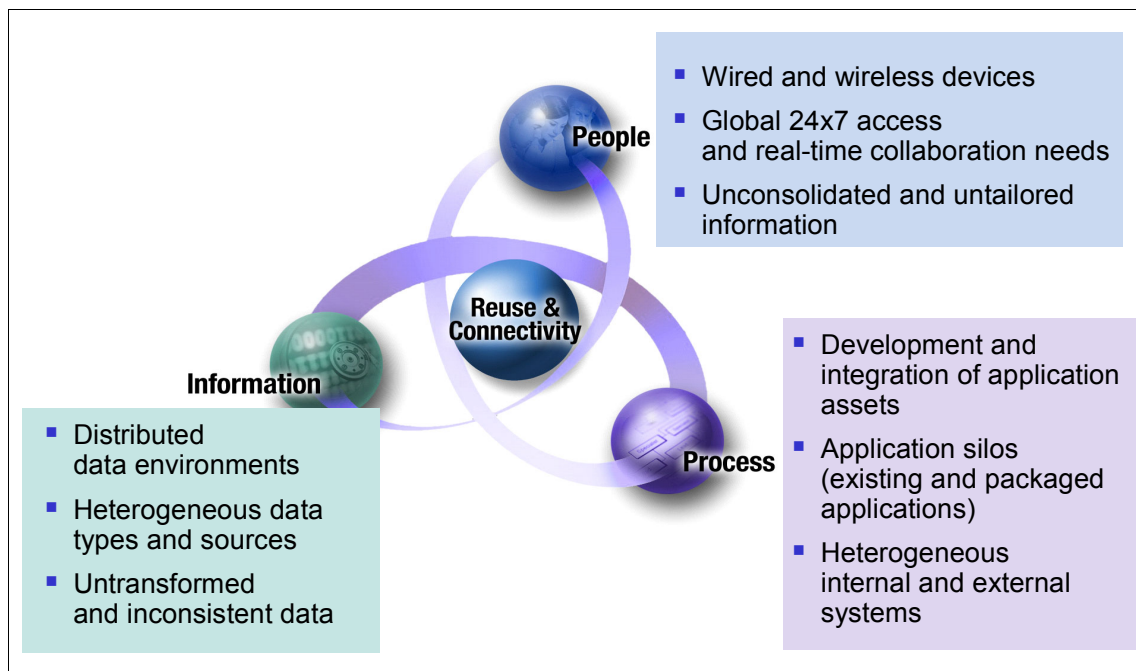


Figure A-10 SOA connects people, processes, and information

Through business-centric SOA, companies can tie IT projects to their business needs directly by addressing the companies immediate pain points.

► **People: Productivity through collaboration**

Improve people productivity by aggregating views that deliver information and interaction in the context of a business process. This enables human and process interaction with consistent levels of service.

Start by building a view of a key business process by aggregating information to help people make better decisions. The next steps are tighter management of performance with alert-driven dashboards that link to more processes.

- ▶ **Process: Business process management for continuous innovation**

Deploy innovative business models quickly with reusable and optimized processes to adapt the enterprise to changing opportunities and threats.

Start by modeling an under-performing process, remove bottlenecks, then simulate and deploy the optimized process. Next, create flexible linkages between multiple processes across the enterprise and outside the firewall to suppliers and partners. Then, monitor the process to measure and track performance.

- ▶ **Information: Delivering information as a service**

Improve business insight and reduce risk with trusted information services delivered in-line and in context.

Start by discovering and understanding information sources, relationships, and the business context. The next steps are to expand the volume and scope of the information delivered as a service across internal and external processes.

The IT-centric entry points to help the enterprise integrate the business-centric SOA entry points are as follows:

- ▶ **Connectivity: Underlying connectivity to enable business centric SOA**

Connectivity has always been a key requirement. SOA brings new levels of flexibility. As well as acting as a building block for additional SOA initiatives, connectivity provided through SOA has distinct, stand-alone value.

- ▶ **Reuse: Create flexible, service base business applications**

Cut costs, reduce cycle times and expand access to core applications through reuse. Analysts estimate it is up to five times less expensive to reuse existing applications than to write new applications.

Use portfolio management to consider which assets you need to run your company. Identify high-value existing IT assets and service-enable them for reuse. Satisfy remaining business needs by creating new services. Finally, create a service registry and repository to provide centralized access and control of these reusable services.

IBM SOA Foundation

The IBM SOA Foundation is an integrated, open standards based set IBM software, best practices and patterns to provide you with the architecture knowledge to get started with SOA. The key elements of the IBM SOA Foundation are the SOA life cycle (model, assemble, deploy, and manage), reference architecture, programming model, and SOA scenarios.

The SOA Foundation scenarios (or simply SOA scenarios) are representative of common scenarios of use of IBM products and solutions for SOA engagements. The SOA scenarios quickly communicate the business value, architecture, and IBM open standards-based software used within the SOA scenario. The concept of realizations are used to provide more specific solution patterns and IBM product mappings within the SOA scenarios.

The SOA scenarios can be used as a reference architecture implementation (starting point) to accelerate the SOA architecture and implementation of your scenario. The SOA scenarios can be implemented using an incremental SOA adoption approach, whereby a customer can incrementally add elements of other SOA scenarios to the environment to achieve their business objectives.

Web services and SOA

This section describes the core technologies of Web services, as well as how Web services are used to implement an SOA.

Note: For more detailed information about Web services, we recommend you read *WebSphere Version 6 Web Services Handbook Development and Deployment*, SG24-6461.

Web services technologies

Web services technology is a collection of standards (or emerging standards) that can be used to implement an SOA. Web services technology is vendor- and platform-neutral, interoperable, and supported by many vendors today.

Web services are self-contained, modular applications, that can be described, published, located, and invoked over networks. Web services encapsulate business functions, ranging from a simple request-reply to full business process interactions. The services can be new or wrap around existing applications.

Core elements

The following are the core technologies used for Web services.

▶ Extensible Markup Language (XML)

XML is the markup language that underlies most of the specifications used for Web services. XML is a generic language that can be used to describe the content in a structured way, separated from its presentation to a specific device.

▶ Simple Object Access Protocol (SOAP)

SOAP is a specification for the exchange of structured XML-based messages between the service provider, service consumer, and service registry, consisting of three parts:

- The format of a SOAP message is an envelope containing zero or more headers and exactly one body. The *envelope* is the top element of the XML document, providing a container for control information, the addressee of a message, and the message itself. *Headers* contain control information, such as quality-of-service attributes. The *body* contains the message identification and its parameters.
- Encoding rules are used for expressing instances of application-defined data types. SOAP defines a programming language independent data type schema based on an XML Schema Descriptor (XSD), plus encoding rules for all data types defined to this model.
- RPC representation is the convention for representing remote procedure calls (RPC) and responses.

SOAP, in principle, is a protocol-independent transport. Consequently, it can potentially be used in combination with a variety of protocols such as HTTP, JMS, SMTP, or FTP. Currently, the most common way of exchanging SOAP messages is through HTTP, which is also the only protocol supported by WS-I Basic Profile 1.0.

▶ Web Services Description Language (WSDL)

WSDL is an XML-based interface and implementation description language. A WSDL document contains the following main elements:

- *Types* is the container for data type definitions using a type system, such as an XML Schema.
- An abstract, typed definition of the data being communicated, a *message* can have one or more typed parts.
- A *port type* is an abstract set of one or more operations supported by one or more ports.

- An *operation* is an abstract description of an action supported by the service that define the input and output message and optional fault message.
 - The *binding* is a concrete protocol and data format specification for a particular port type. The binding information contains the protocol name, the invocation style, a service ID, and the encoding for each operation.
 - The *service* as a collection of related ports.
 - The *port* is a single endpoint, an aggregation of a binding and a network address.
- Universal Description, Discovery, and Integration (UDDI)
- UDDI is both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information about service providers and Web services.

Standards

Figure A-11 on page 341 displays a stacked view of Web services technologies. Most of the technologies displayed have been standardized. Since interoperability is a key goal of Web services, an open industry organization known as the *Web services Interoperability Organization (WS-I)* has been created to allow interested parties such as IBM and Microsoft to work together to maximize interoperability between Web services implementations. For more information about WS-I, visit their Web site:

<http://ws-i.org>

Note: Web services standards are evolving at a rapid pace. For the most current information, we recommend that you reference the Web services standards information online at sites such as IBM developerWorks:

<http://www.ibm.com/developerworks/webservices/standards/>

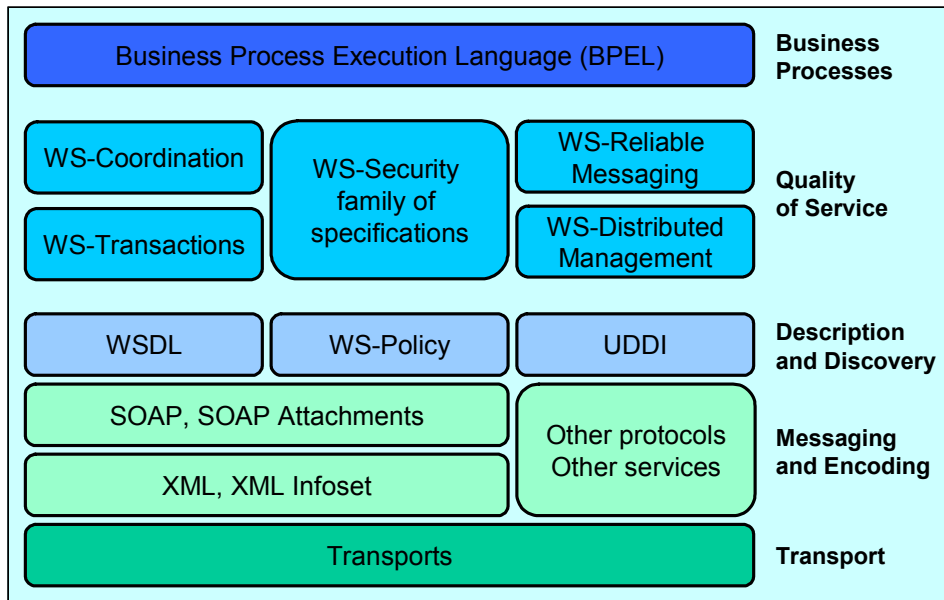


Figure A-11 Stack view of Web services technology

Web services for J2EE

Web services for J2EE V1.1 (WSEE) support is included in the J2EE V1.4 specification, which is used by WebSphere Application Server V6. The Java API for XML-based RPC (JAX-RPC) provides the programming model for SOAP-based applications by abstracting the runtime details and providing mapping services between Java and WSDL.

Exposing Web services

The port component is fundamental part of a Web service used to define the programming model artifacts that make the Web service portable. The programming model includes:

- ▶ A *WSDL definition* provides the description of a Web service.
- ▶ The *service endpoint interface* (SEI) defines the operations of the Web service.
- ▶ A *service implementation bean* implements the SEI methods to provide the business logic of the Web service.
- ▶ The *security role references* provide instance-level security checks across different modules.

From a server programming model perspective, there are primarily two types of J2EE application artifacts exposed as Web services (service provider):

- ▶ Stateless session EJB (EJB container)
- ▶ JAX-RPC servlet-based service that invokes a JavaBean, known as a service endpoint (Web container)

There are three principal approaches to generating a Web service, depending on the elements that are used to start the creation of the service:

- ▶ An existing application is used to generate the Web service, which includes a service wrapper used to expose application functionality. This is known as the *bottom-up* approach.
- ▶ An existing service definition WSDL is used to generate a new application for a specific programming language and model. This is known as the *top-down* approach.
- ▶ An existing group of already generated Web services provides a new combination of functionality (multiple services). Composing a new Web service in this way might include the use of workflow technologies.

Invoking Web services

The J2EE client container provides the WSEE runtime used by a Web services client application, to access and invoke Web service methods. The J2EE client container is responsible for the JNDI name to service implementation mapping.

From a client application programming perspective, there are three mechanisms used to invoke a Web service (service consumer) from the Web service client application:

- ▶ A *static stub* is created before being deployed to the runtime environment. This requires complete knowledge of the WSDL.
- ▶ A *dynamic proxy* class is created during runtime. Only a partial WSDL definition is required (port type and bindings).
- ▶ A *dynamic invocation* interface does not require WSDL knowledge. The signature or service name are unknown until runtime.

The task to build or generate a Web service client (service consumer) depends on the methods of how the client is binding to a Web service server. The client uses a local service stub or proxy to access the remote server and service. The WSDL document is used to generate or set up the particular stub or proxy. The stub or proxy knows at request time how to invoke the Web service based on the binding information.

Web services and SOA

Web services technology is a collection of standards (or emerging standards) that can be used to implement a service-oriented architecture (SOA). That is not to say that Web services and SOA are intrinsically linked, because they can be implemented separately.

In fact, many significant SOAs are proprietary or customized implementations that are based on reliable messaging and Enterprise Application Integration middle ware (for example, IBM WebSphere MQ and IBM WebSphere Business Integration Message Broker) do not use Web services technologies. Also, most existing Web services implementations consist of point-to-point integrations that address a limited set of business functions between a defined set of cooperating partners.

The logical links between Web services and SOA are:

- ▶ Web services provide an open-standard model for creating explicit, implementation-independent descriptions of service interfaces.
- ▶ Web services provide communication mechanisms that are location-transparent and interoperable.
- ▶ Web services are evolving, through Business Process Execution Language for Web Services (WS-BPEL), document-style SOAP, Web services Definition Language (WSDL), to support the implementation of well-designed services that encapsulate and model reusable function in a flexible manner.



B

IBM SOA Foundation

If you are not familiar with the concepts of SOA, read Appendix A, “Introduction to service-oriented architecture” on page 323 before reading this appendix.

This chapter discusses the *IBM SOA Foundation*, an integrated, open standards based set of IBM software, best practices, and patterns designed to provide what you need to get started with SOA from an architecture perspective.

The key elements that we discuss in this chapter are:

- ▶ SOA life cycle (model, assemble, deploy, and manage)
- ▶ Reference architecture
- ▶ SOA scenarios overviews for the rest of this IBM Redbook

SOA Foundation overview

The SOA Foundation scenarios (or simply SOA scenarios) are representative of common scenarios of use of IBM products and solutions for SOA engagements. The SOA scenarios communicate the business value, architecture, and IBM open standards-based software used within the SOA scenario.

The SOA scenarios can be used as a reference architecture (starting point) to accelerate the SOA architecture and implementation of your customer scenario. The SOA scenarios can be implemented using an incremental SOA adoption approach, whereby a customer can incrementally add elements of other SOA scenarios to the environment to achieve their business objectives.

To gain a better understanding of the IBM SOA Foundation, we explore the following defining elements:

- ▶ SOA Foundation life cycle
- ▶ SOA Foundation Reference Architecture
- ▶ SOA Foundation scenarios

Note: For a more detailed explanation of the SOA Foundation, refer to *IBM SOA Foundation, An Architectural Introduction and Overview V1.0*, found at:

<http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf>

SOA Foundation life cycle

IBM customers have indicated that they think of SOA in terms of a life cycle. As seen in Figure B-1 on page 347, the IBM SOA Foundation includes the following life cycle phases:

- ▶ Model
- ▶ Assemble
- ▶ Deploy
- ▶ Manage

There are a couple of key points to consider about the SOA life cycle:

First, the SOA life cycle phases apply to all SOA projects. Second, the activities in any part of the SOA life cycle can vary in scale and the level of tooling used depending on the stage of adoption.

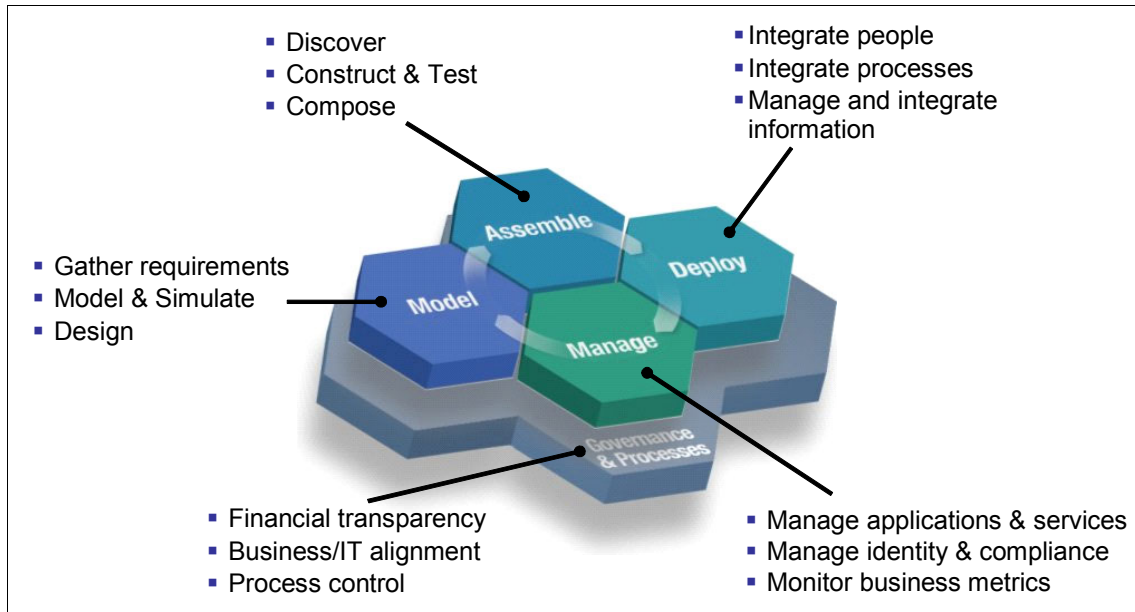


Figure B-1 IBM SOA Foundation life cycle

Model

Modeling is the process of capturing the business design from an understanding of business requirements and objectives. The business requirements are translated into a specification of business processes, goals, and assumptions for creating a model of the business. Many businesses do not go through a formal modeling exercise. In some case, businesses that do perform modeling use primitive techniques such as drawing the design in Visio® or using text documents.

Capturing the business design using a sophisticated approach that includes the use of specialized tooling lets you perform what-if scenarios with various parameters the business might experience. The process can then be simulated using those parameters to predict the effect that process will have on the business and IT systems. If the achieved results do not match the business objectives, then the process definition can be refined.

The model also captures key performance indicators that are important measurements of your business, such as business metrics. For example, these measurements could include a measure of the new accounts that you have opened in a given month. These key performance indicators are built into the assembly of the application. In addition, you can monitor the indicators in production, capturing critical data to measure if the objectives are being met.

Assemble

The business design is used to communicate the business objectives to the IT organization that will assemble the information system to implement the design. The enterprise architect works closely with the business analyst to convert the business design into a set of business process definitions, as well as activities used to derive the required services from the activity definitions. The enterprise architect and business analyst work with the software architect to fully develop the design of the services.

While you are resolving the design and implementation of the modeled business processes and services, you should perform a search of existing artifacts and applications to find components that meet the design requirements. Some applications will fit perfectly. Some applications will have to be refactored, and some will have to be augmented to meet the design requirements.

These existing assets should be rendered as services for assembly into composite applications. Any new services required by the business design must be created. Software developers should use the SOA programming model to create these new services.

Lastly, the assemble phase includes applying a set of policies and conditions to control how your applications operate in the production runtime environment. For example, these policies and conditions include business and government regulations. In addition, the assemble phase includes critical operational characteristics such as packaging deployment artifacts, localization constraints, resource dependency, integrity control, and access protection.

Deploy

The deploy phase of the life cycle includes a combination of creating the hosting environment for the applications and the deployment tasks of those applications. This includes resolving the application's resource dependencies, operational conditions, capacity requirements, and integrity and access constraints.

A number of concerns are relevant to construction of the hosting environment, including the presence of the already existing hosting infrastructure supporting applications and pre-existing services. Beyond that, you must consider appropriate platform offerings for hosting the user interaction logic, business process flows, business-services, access services, and information logic.

Manage

The manage phase includes the tasks, technology and software used to manage and monitor the services and business processes that are deployed to the production runtime environment.

Monitoring is a critical part of ensuring the underlying IT systems and application are up and running to maintain service availability requirements. Monitoring includes these activities:

- ▶ Monitoring performance of service requests and timeliness of service responses
- ▶ Maintaining problem logs to detect failures in various services and system components, as well as localizing failures and restoring the operational state of the system

Managing the system also involves performing routine maintenance, administering and securing applications, resources and users, and predicting future capacity growth to ensure that resources are available when the demands of the business warrant using them. The security domain includes such topics as authentication, single sign-on, authorization, federated identity management, and user provisioning.

The manage phase also includes managing the business model, and tuning the operational environment to meet the business objectives expressed in the business design, and measuring success or failure to meet those objectives. SOA is distinguished from other styles of enterprise architecture by its correlation between the business design and the software that implements that design, and its use of policy to express the operational requirements of the business services and processes that codify the business design. The manage phase of the life cycle is directly responsible for ensuring those policies are being enforced, and for relating issues with that enforcement back to the business design.

Governance

SOA Governance is critical to the success of any SOA project. Governance helps clients extend the planned SOA across the enterprise in a controlled manner. SOA Governance has four core objectives or challenges:

- ▶ Establish decision rights.
- ▶ Define high value business services.
- ▶ Manage the life cycle of your assets.
- ▶ Measure effectiveness.

SOA Foundation Reference Architecture

This section describes the SOA Foundation Reference Architecture, which includes the components and middle ware services used by applications in the runtime environment.

Figure B-2 depicts the SOA Foundation Reference Architecture solution view used to decompose an SOA design. SOA puts a premium on the role of the Enterprise Architect, who is responsible for spanning between the business design and the information system that codifies that design.

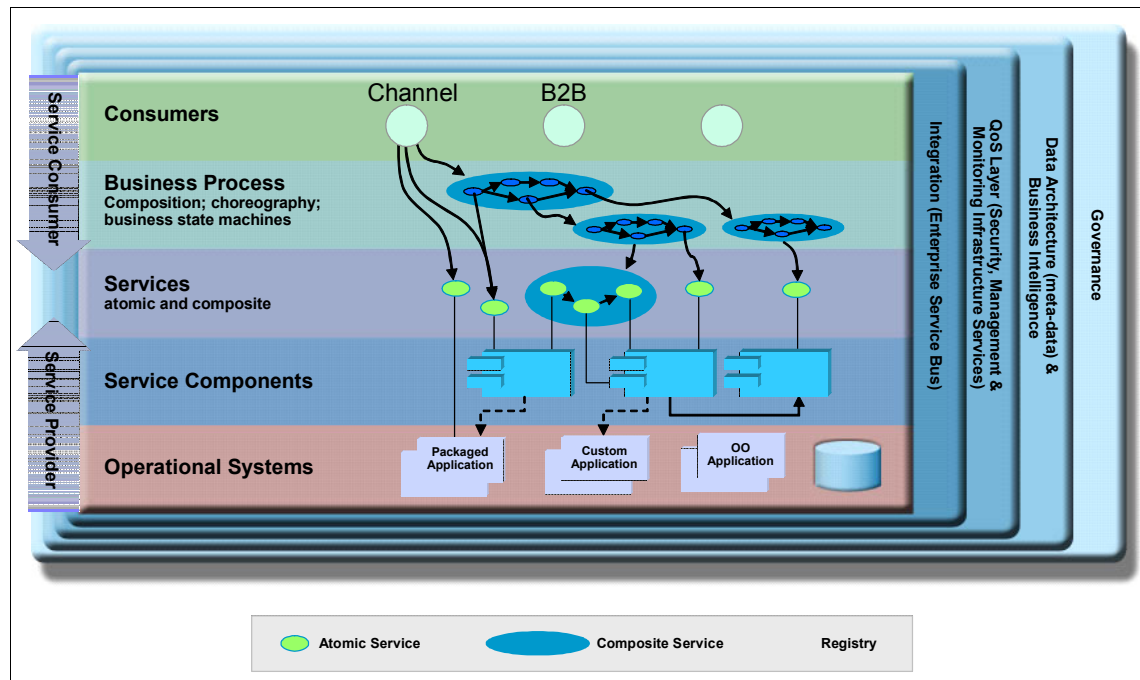


Figure B-2 SOA Foundation Reference Architecture: Solution view

While this flow describes a top-down approach, flow variations include a bottom-up approach, and the more common meet-in-the-middle approach. When taking a top-down approach, the enterprise architect starts by identifying the business processes and business services used by the business users. The business users are consumers of the processes and services. Business processes are treated as compositions of other business processes and services, and therefore should be separated into their subordinate subprocesses and services.

Services and business processes are then detailed into service components. Service components include a detailed set of definition metadata used to describe the service to the information system. Services can be aggregated into module assemblies. The module assemblies are used to establish related design concerns, and begin the planning to determine what teams will collaborate to implement the related services to be deployed as a single unit.

The resulting set of business process definitions, services, and schemas make up the logical architecture of the application. The enterprise architect then needs to map that logical architecture to a physical architecture.

We have included a summary description for each of the services found in the logical architecture displayed in Figure B-3. The services found in the center (Interaction, Process, Information, Partner, Business Application, and Access) are the core set of services used by application within the runtime environment when deployed. The other outer services displayed are used in support of the core services.

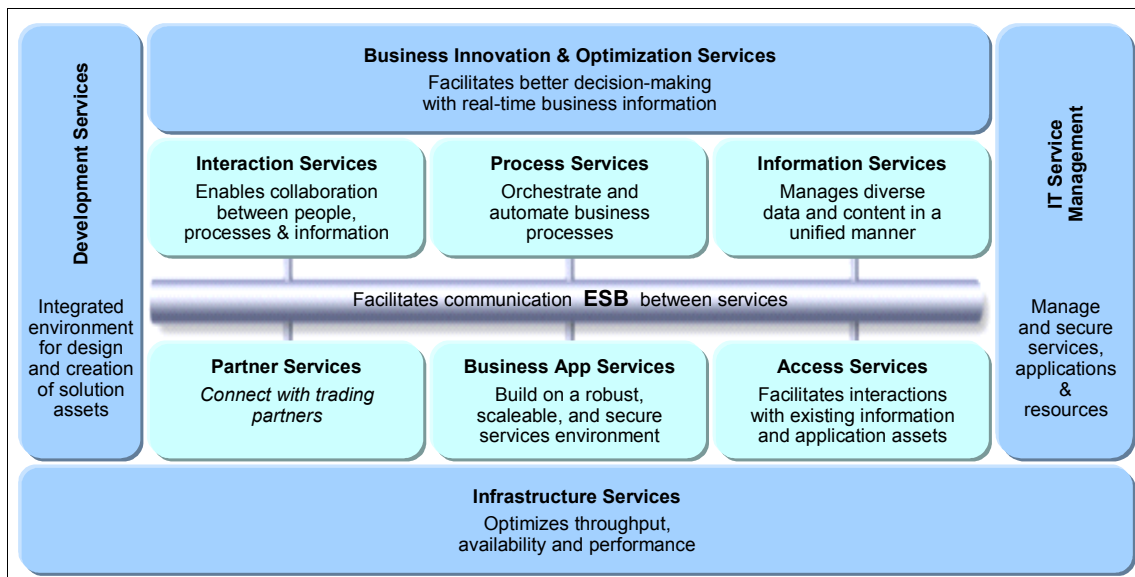


Figure B-3 SOA Foundation Reference Architecture: Middleware Services view

Core components of the logical architecture

This section includes a brief description on the following core components of the logical architecture:

- ▶ Interaction services
- ▶ Process services
- ▶ Business application services
- ▶ Information services
- ▶ Access services
- ▶ Partner services

Interaction services

Interaction services provide the capabilities required to deliver IT functions and data to users, meeting their specific preferences.

Process services

Process services provide the control capabilities required to manage the flow and interactions of multiple services in ways that implement business processes.

Business application services

Business application services are called by service consumers. Service consumers include other components in the logical architecture such as portal or a business processes.

Information services

Information services provide the capabilities necessary to federate, replicate, and transform disparate data sources.

Access services

Access services provide bridging capabilities between core applications, prepackaged applications, enterprise data stores, and the ESB to incorporate services that are delivered through existing applications into an SOA.

Partner services

Partner services provide the document, protocol, and partner management capabilities for business processes that involve interactions with outside partners and suppliers.

Supporting components of the logical architecture

This section includes a brief description of the supporting components of the SOA Foundation logical architecture used in support of the core components:

- ▶ Enterprise Service Bus
- ▶ Business innovation and optimization services
- ▶ Development services
- ▶ IT service management
- ▶ Infrastructure services

Enterprise Service Bus

The Enterprise Service Bus (ESB) or simply bus, provides an infrastructure that removes the direct connection dependency between service consumers and providers. Consumers connect to the bus and not the provider that actually implements the service. This type of connection further decouples the consumer from the provider. A bus also implements further value add capabilities, such as security and delivery assurance. It is preferred to implement these capabilities centrally within the bus at an infrastructure level rather than within the application. The primary driver for an ESB, however, is that it increases decoupling between service consumers and providers.

Although it is relatively straight forward to build a direct link between a consumer and provider, these links can lead to an interaction pattern that consists of building multiple point-to-point links that perform specific interactions. With a large number of interfaces, this quickly leads to the build up of a complex spaghetti of links with multiple security and transaction models. When routing control is distributed throughout the infrastructure, there is typically no consistent approach to logging, monitoring, or systems management. This type of environment is difficult to manage or maintain and inhibits change.

Note: An ESB can be thought of as an architectural pattern, with an implementation to match the deployment needs. There are two IBM ESB products:

- ▶ IBM WebSphere Enterprise Service Bus
- ▶ IBM WebSphere Message Broker

In addition, there are a number of products that extend the capabilities of these ESBs, including DataPower XML Security Gateway XS40.

Business innovation and optimization services

Business innovation and optimization services are primarily used to represent the tools and the metadata structures for encoding the business design, including the business policies and objectives.

Business innovation and optimization services exist in the architecture to help capture, encode, analyze, and iteratively refine the business design. The services also include tools to help simulate the business design. The results are used to predict the effect of the design, including the changes the design will have on the business.

Development services

Development services encompass the entire suite of architecture tools, development tools, visual composition tools, assembly tools, methodologies, debugging aids, instrumentation tools, asset repositories, discovery agents, and publishing mechanisms needed to construct an SOA based application.

IT service management

After the application has been deployed to the runtime environment, it must be managed along with the IT infrastructure on which it is hosted. IT service management represents the set of management tools used to monitor your service flows, the health of the underlying system, the utilization of resources, the identification of outages and bottlenecks, the attainment of service goals, the enforcement of administrative policies, and recovery from failures.

Infrastructure services

Infrastructure services form the core of the information technology runtime environment used for hosting SOA applications. These services provide the ability to optimize throughput, availability, performance, and management.

SOA Foundation scenarios

The SOA Foundation scenarios (simply SOA scenarios) are representative of common scenarios of IBM products and solutions for SOA engagements. The SOA scenarios quickly communicate the business value, architecture, and IBM open standards-based software used within the SOA scenario. The SOA scenarios can be implemented as part of an incremental adoption of SOA growing from one scenario to using elements of multiple scenarios together. The concept of realizations are used to provide more specific solution patterns and IBM product mappings within the SOA scenarios.

The SOA scenarios can be used as a reference architecture implementation (starting point) to accelerate the SOA architecture and implementation of your customer scenario.

Figure B-4 on page 355 displays the SOA scenarios (Service Creation, Service Connectivity, Interaction and Collaboration Services, Business Process

Management, and Information as a Service), and the relationship between the scenarios.

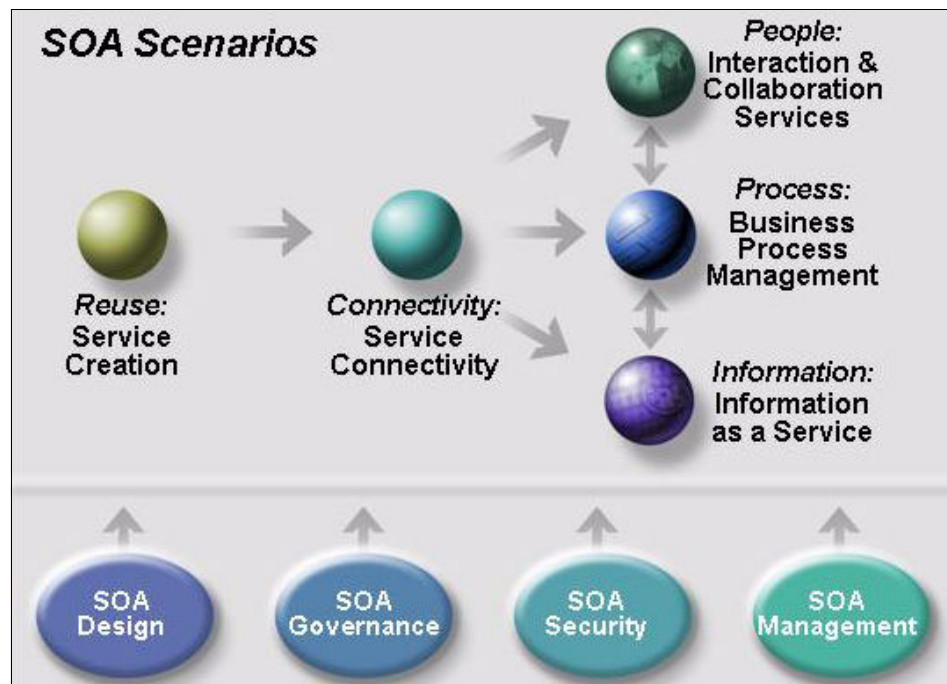


Figure B-4 SOA scenarios and entry points

We have included examples of how the scenarios can be used together and adopted incrementally. For example, it is common that the other scenarios will include service creation and often want connectivity. In addition, the scenarios can be used together, such as a portal accessing a business process or a portal accessing an information service through an ESB from a service consumer.

SOA Design, SOA Governance, SOA Security, and SOA Management are used in each of the SOA scenarios based on customer requirements.

SOA Design is focused on service modeling and service design. The service model is an abstraction of the IT services implemented within an enterprise and supporting the development of one or more service-oriented solutions. It is used to conceive and document the design of the software services. It is a comprehensive, composite work product encompassing all services, providers, specifications, messages, collaborations, and relationships between them.

The service design model extends the standard Rational Unified Process (RUP) design model by adding the service components. The RUP for SOA service design includes service identification and modeling techniques derived from IBM

Service-Oriented Modeling and Architecture (SOMA). The service components artifact is intended for use in describing the realization of a service specification. A service component can provide the realization for one or more services by the realization of multiple service specifications. The set of model elements on the inside of the component represent the concrete realization of the structural, behavioral, and policy contract described by these service specifications.

SOA Governance is critical to the success of any SOA project. Governance helps clients extend the planned SOA across the enterprise in a controlled manner. SOA Governance has four core objectives or challenges:

- ▶ Establish decision rights
- ▶ Define high value business services
- ▶ Manage the life cycle of assets
- ▶ Measure effectiveness

Every SOA implementation encompasses the need of a security architecture that enables secure business transactions across and within enterprises. The *SOA Security* reference architecture includes security services such as identity services, authentication services, authorization and privacy services, message protection services, and audit services. Other important factors of this reference architecture include security policy and security management infrastructures. For more detailed information about the SOA security reference architecture, refer to the IBM Redbook *Understanding SOA Security Design and Implementation*, SG24-7310.

SOA-based composite applications span the architectural layers of the SOA Foundation Reference Architecture Solution View (see Figure B-2 on page 350). SOA composite applications require a mind shift from a silos-based, application-management approach for two key reasons:

- ▶ There is a need for a management approach and tooling that covers the end-to-end view of transactional performance and availability of the composite application.
- ▶ Second, it is important to understand the relationship of service consumers and providers for composite applications. The importance of monitoring the availability and performance of common services increases when reused by service consumers that depend on this functionality.

SOA Management provides best practices and software for managing and monitoring composite applications and supporting infrastructure across the architectural layers. This includes managing and monitoring the services layer (providers and consumers), transactional performance, middle ware, and operational systems with specific metrics to ensure Service Level Agreements (SLAs) meet the defined business objectives.

We have provided a summary for each of the following SOA scenarios:

- ▶ Service Creation scenario
- ▶ Service Connectivity scenario
- ▶ Interaction and Collaboration Services scenario
- ▶ Business Process Management scenario
- ▶ Information as a Service scenario

Service Creation scenario

The Service Creation scenario (service provider and service consumer) is used to demonstrate exposing application functionality of an existing application or new business logic as a service. The services can then be consumed by other services or client applications within and between enterprises. The key driver for this scenario is the reuse of existing or new application functions as services.

There are many possible examples to illustrate the Service Creation scenario. We have included a summary of the following four common realizations of the Service Creation scenario:

- ▶ Directly expose existing applications as services
- ▶ Indirectly expose existing applications with service
- ▶ Create an EJB Web service from WSDL
- ▶ Consume services from third-party service providers

One of the goals of enterprise transformation is to enable access to Enterprise Information System (EIS) applications as services, with the objective of leveraging the investment of existing business applications and systems. By adopting an SOA approach, the EIS application functionality can be turned into reusable services that can be consumed by a new set of client applications and users.

The Service Creation scenario includes two methods of exposing EIS applications as services, known as direct and indirect exposure. We use CICS as an example of an existing EIS application for both the direct and indirect exposure realization examples.

Directly expose existing applications as services

In this realization, we expose existing applications directly as services. The existing application can be a wide range of application types, such as an EIS (for example, CICS or IMS), J2EE application, SAP®, and so forth. A key distinction for this realization is that the service interface to be exposed is defined by the existing application. This approach to creating a service is known as *bottom-up*.

We use CICS as an example of an existing EIS application. In this realization example, we access applications hosted by CICS Transaction Server *directly* as Web services.

The core IBM products used for this example realization are:

- ▶ Assemble: WebSphere Developer for IBM System z™ V6.0.1
- ▶ Deploy:
 - CICS Transaction Server V3.1
 - WebSphere MQ V6

Additionally, the following IBM products can be considered, depending on specific customer requirements:

- ▶ Manage: Tivoli OMEGAMON® XE for CICS V3.10
- ▶ Governance: WebSphere Service Registry and Repository

WebSphere Service Registry and Repository: IBM currently intends to make available in the second half of 2006 a WebSphere service registry and repository capability that will allow customers to securely register business services for finding, publishing, and notifying changes to SOA infrastructure components, such as enterprise service bus and process servers. Customers will also be able to house the metadata about business services in managing the life cycle of a service in SOA. The new capabilities will also include a model for governance to provide guidance and oversight for a SOA project.

While this is our intention, all statements regarding IBM plans, directions, and intent are subject to change or withdrawal without notice.

Indirectly expose existing applications with service components

For this realization, we expose existing application functionality indirectly with service components. In this realization, business alignment is achieved by defining the service interface (WSDL) independently of existing assets. This approach to creating a service is known as top-down. In practice, the implementation of the service might be more completely described as a combination of top-down and meet-in-the-middle.

We use CICS as an example of an existing EIS. In this realization example, we expose existing COMMAREA applications hosted by CICS Transaction Server *indirectly* by creating middle-tier Web services to access CICS. The middle-tier Web service wraps a session EJB that uses the CICS ECI resource adapter to communicate with the CICS Transaction Gateway (CTG) to access CICS Transaction Server. The CICS ECI resource adapter is a JCA adapter for WebSphere Application Server packaged with the CTG.

The core IBM products used for this example realization are as follows:

- ▶ Assemble: Rational Application Developer V6.0.1
- ▶ Deploy:
 - WebSphere Application Server V6 (IBM System z V6 or distributed platform depending on the selected runtime topology).
 - CICS Transaction Gateway (TG) V6.1 (CICS ECI resource adapter for WebSphere is included with the CICS TG.)
 - CICS V2.x

Additionally, the following IBM products might be considered, depending on specific customer requirements:

- ▶ Model: Rational Software Architect V6
- ▶ Manage:
 - Tivoli OMEGAMON-XE for CICS V3.1
- ▶ Governance: WebSphere Service Registry and Repository

Create an EJB Web service from WSDL

In this realization, we create a new session EJB Web service from an existing service WSDL. This realization is also known as *create from scratch*. This realization uses a top-down approach to creating a Web Service.

The core IBM products used for this realization are:

- ▶ Assemble: Rational Application Developer V6
- ▶ Deploy: WebSphere Application Server Network Deployment V6
- ▶ Manage: Tivoli Composite Application Manager for SOA V6

Additionally, the following IBM products might be considered, depending on specific customer requirements:

- ▶ Model: Rational Software Architect V6
- ▶ Deploy: WebSphere Service Registry and Repository
- ▶ Manage: Tivoli Composite Application Manager for WebSphere V6
- ▶ Governance: WebSphere Service Registry and Repository

Consume services from third-party service providers

In this realization, client applications consume services from third-party service providers. This realization represents the view of a consumer that is using one or more third-party services. The service consumer only sees the service interfaces. The endpoints and any security or transport constraints are imposed by the service provider.

For example, a Web services client application can invoke an address verification Web service from a third-party service provider.

This realization assumes that the WSDL is WS-I compliant and JAX-RPC compatible. The service provider is outside the enterprise firewall and security is implemented between the consumer and provider using mutual SSL authentication.

The core IBM products used for this realization as follows:

- ▶ Assemble: Rational Application Developer V6
- ▶ Deploy: WebSphere Application Server V6

Additionally, the following IBM products might be considered, depending on specific customer requirements:

- ▶ Manage:
 - Tivoli Composite Application Manager for SOA V6
 - Tivoli Composite Application Manager for WebSphere V6
- ▶ Governance: WebSphere Service Registry and Repository

Service Connectivity scenario

The Service Connectivity scenario is used to demonstrate the integration of service providers and consumers, allowing for the reuse of existing and new services across multiple channels. This scenario is appropriate for an enterprise that has a set of core services or systems that are to be made available as services to internal and external clients. Flexibility to make changes to service providers and service clients independent from each other is a requirement.

The focus of this scenario is on the underlying connectivity used to support business-centric SOA. An enterprise service bus provides decoupling between clients and providers, providing the flexibility to implement applications more quickly. In circumstances where services are provided to or consumed from a third party, an ESB gateway can be used in conjunction with the ESB to add security measures. An ESB gateway alone can be sufficient if all of your service interactions are with third parties and you have the basic requirements to mediate between service consumers and providers.

Note: A more detailed description of the Service Connectivity scenario can be found in *Patterns: SOA Foundation Service Connectivity Scenario*, SG24-7228.

Implementations of this scenario have the following features:

- ▶ Enables changes to the implementation of a service without affecting clients.

- ▶ Registers services to a service registry.
- ▶ Uses an enterprise service bus as the integration point between service providers and service consumers.
- ▶ Enables clients to access a service with a different interface and protocol than what the service consumer supports.
- ▶ Uses an ESB gateway to isolate and protect services.
- ▶ Enables management and monitoring of services to insure Service Level Agreements.
- ▶ Provides security and credential mapping (where needed) to ensure proper use of the services.

Specific connectivity and integration requirements for an enterprise will ultimately drive product selection of the ESB and supporting products. The choice of runtime products can include one or more of the following:

- ▶ WebSphere Message Broker
- ▶ WebSphere Enterprise Service Bus
- ▶ IBM DataPower SOA Appliances
- ▶ Web Services Gateway (WebSphere Application Server Network Deployment V6)
- ▶ WebSphere Adapters
- ▶ WebSphere Service Registry and Repository

The choice of SOA life cycle products will depend largely on the runtime products selected. The following products can be used to support the runtime environment:

- ▶ WebSphere Message Broker Toolkit
- ▶ Rational Application Developer
- ▶ WebSphere Integration Developer
- ▶ IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA)
- ▶ IBM Tivoli Composite Application Manager for WebSphere (ITCAM for WebSphere)
- ▶ IBM Tivoli Composite Application Manager for RTT V6.0 (TCAM for RTT)
- ▶ OMEGAMON for Messaging
- ▶ Tivoli Access Manager
- ▶ Tivoli Federated Identity Manager

Realizations have been developed that will help you understand how the scenario can be used and how products are selected. A *realization* is an example business case that describes a customer situation and the solution. We have included a summary of the following common realizations of the Service Connectivity scenario:

- ▶ Gateway: For use when interactions with third parties are present and mediation requirements are basic.
- ▶ Local integration: For use with standards-based interactions that require routing capabilities.
- ▶ Web services access to Enterprise Information Systems: For use when requiring access to EIS systems.
- ▶ Expose existing systems to heterogeneous clients: For use in a diverse, nonstandards-based environment.

Gateway

An ESB gateway can be used alone or in conjunction with an ESB to provide controlled and secure service interaction between internal or external domain boundaries. In this realization, its primary function is to provide secure access to resources when interacting with third parties. An ESB gateway can also provide basic functionality, such as protocol switching and message switching, to enable interaction between service consumers and service providers.

This realization assumes the customer has adopted standards-based technology, has an existing infrastructure, and has the following business requirements:

- ▶ Standards-based consumers/providers will use SOAP/HTTP for transport.
- ▶ Dynamically add new providers and consumers at runtime.
- ▶ Support a defined, high response time with a moderate load.
- ▶ SOA security for interaction with consumers and providers. Security may need to be adapted between the consumers and providers.
- ▶ Requests and responses must be logged to a file

The following technical requirements have been identified:

- ▶ Many services deployed will require the same mediation flow. An ESB gateway will minimize administration and streamline the process for making new services available.
- ▶ Services must be monitored for performance and usage.
- ▶ Monitoring for all components must be integrated into existing management infrastructure.

The IBM products used for this realization are as follows:

- ▶ **Deploy:** IBM DataPower XML Security Gateway XS40

The customer chooses the DataPower XS40 model for the runtime. The XS40 is designed specifically to provide XML acceleration and SOA security and can provide the basic mediation functions required. Because the requirements for mediating the interaction between consumers and providers is met, the XS40 as an ESB gateway is sufficient and no ESB product is required.

- ▶ **Assemble:** DataPower Toolkit

- ▶ **Manage:** IBM Tivoli Composite Application Manager for SOA

ITCAM for SOA will be used to monitor Web services flowing through the DataPower appliance.

- ▶ **Manage:** Tivoli Access Manager

The XS40 can be integrated with Tivoli Access Manager to secure applications.

Local integration

A local integration solution provides multi-channel access for clients to an existing service with a range of connectivity options for standards-based clients and services, message routing with or without the use of external data, message transformation and augmentation, protocol switching, and security.

With this type of connectivity, a client can request a secure service without knowledge of its location. Transparent to the client, requests can be routed to the service that can best handle the request. Also transparent to the client is the message format and transport protocol required to access the provider. The response could be immediate or delayed.

This realization assumes the customer has adopted standards-based technology, has an existing WebSphere Application Server infrastructure, and has the following business requirements:

- ▶ Provide integration of multiple client channels to service providers.
- ▶ Provide routing of client requests to the appropriate service provider.
- ▶ Intranet environment that does not require WS-Security or other complex security considerations.
- ▶ Support moderate volume of requests.

The following technical requirements have been identified:

- ▶ Message data from clients must be examined in order to determine the service provider to route the request to.
- ▶ Clients and service providers will use JMS, SOAP/JMS, or SOAP/HTTP.
- ▶ Data transformation will be required. This should be done with XSLT.

The core IBM products used for this realization are as follows:

- ▶ **Deploy: WebSphere Enterprise Service Bus**
WebSphere ESB provides the transport flexibility to support the transports required by the customer. WebSphere ESB also has the mediation capabilities required to perform the message routing and transformation required.
- ▶ **Assemble: WebSphere Integration Developer**
WebSphere Integration Developer is the development and assembly tool for building WebSphere ESB mediations.
- ▶ **Manage: IBM Tivoli Composite Application Manager for SOA**
ITCAM for SOA will be used to monitor Web services requests as they arrive at WebSphere ESB.

Web services access to Enterprise Information Systems

An ESB can be used to provide access to EIS systems through the use of adapters. Mediations in the ESB are used to adapt the client request to a form understood by the adapter, and then to adapt the response to the client's format.

This realization assumes the customer has adopted standards-based technology, has an existing WebSphere Application Server infrastructure, and has the following business requirements:

- ▶ Provides Web service access to functionality in an Enterprise Information System, such as SAP R/3®, PeopleSoft®, or Oracle® Financials.
- ▶ An intranet environment that does not require WS-Security or other complex security considerations.
- ▶ The integration is based on message exchange/data replication scenarios; there is no business process or data synchronization between clients and EIS systems.
- ▶ Supports moderate volume of requests.

The following technical requirements have been identified:

- ▶ The targeted integration is point-to-point, although multiple EISs can be exposed as Web services at the same time.
- ▶ Data transformation will be required. This should be done with XSLT.
- ▶ Log the messages as they flow through the hub (want to log asynchronously to a file).

The IBM products used for this realization are:

- ▶ **Deploy: WebSphere Enterprise Service Bus and WebSphere Adapters**
WebSphere ESB supports the SOAP/HTTP transport required by the customer. WebSphere Adapters provide the EIS adapters required. WebSphere ESB also provides the mediation capability required to do XSLT transformation on the data and includes a logging function to log messages as they flow through the mediation.
- ▶ **Assemble: WebSphere Integration Developer**
WebSphere Integration Developer is the development and assembly tool for building WebSphere ESB mediations. It includes the enterprise discovery capabilities needed to incorporate the WebSphere Adapters into the mediation applications.
- ▶ **Manage: IBM Tivoli Composite Application Manager for SOA**
ITCAM for SOA will be used to monitor Web services requests as they arrive at WebSphere ESB.

Expose existing systems to heterogeneous clients

An integration solution that includes a range of diverse business applications must provide connectivity for a wide range of service consumers and service providers as well as advanced options for message mediation, including message augmentation, message routing, and the ability to decompose messages into multiple requests and to recompose the responses.

This type of connectivity would provide the most advanced options for integrating dissimilar and wide-spread service consumers and service providers. Clients can request a secure service that may be provided by one or more service providers with the service composition occurring within the ESB. Services and clients also have a wide range of connectivity options. Connectivity to existing applications as well as standards-based applications are managed by the ESB.

This realization assumes the customer has extensive existing systems as well as some newer Web services based systems and has the following business requirements:

- ▶ Providers use a variety of heterogeneous protocols.
- ▶ Any provider must be accessible through basic Web services that will be used by a variety of clients.
- ▶ Support moderate volume of requests.
- ▶ An intranet environment does not require SOA security or other complex security considerations.
- ▶ Global transactions across multiple heterogeneous transaction managers for some providers.

The following technical requirements have been identified:

- ▶ The ESB must support communication protocol conversion.
- ▶ The ESB must support flexible data model conversion, with acceptable performance and adequate tooling.
- ▶ Enterprise class persistent messaging backbone.
- ▶ Global transactions management.
- ▶ The ESB must adapt the service definitions between the consumers and providers.

The IBM products used for this realization are as follows:

- ▶ Deploy: WebSphere Message Broker, WebSphere MQ, and WebSphere Adapters

WebSphere Message Broker is selected to provide the ESB capabilities, including mediation support. WebSphere MQ will be used to provide an enterprise class persistent messaging backbone. This combination will support the wide variety of transport protocols and conversions required for the integration solution. WebSphere Adapters provide connectivity to existing systems.

- ▶ Assemble: Message Brokers Toolkit

The Message Brokers Toolkit is the development tool for building mediation message flows in WebSphere Message Broker and provides the runtime configuration and management tools.

Interaction and Collaboration Services scenario

The Interaction and Collaboration Services scenario features single sign-on and a role-based portal used to consolidate access to information and application within the enterprise and between enterprises.

The key drivers for this scenario are to improve people productivity and consumability of applications and content. The content can be personalized in the aggregated portal page based on the user role.

The core IBM products used for the Interaction and Collaboration Services scenario are as follows:

- ▶ Assemble:
 - Rational Application Developer V6
 - Bowstreet Portlet Factory
- ▶ Deploy: WebSphere Portal V5.1
- ▶ Manage: Tivoli Composite Application Manager for SOA V6

Additionally, the following IBM products may be considered by a customer depending on specific requirements:

- ▶ Assemble: WebSphere Integration Developer
- ▶ Deploy:
 - WebSphere Process Server
 - WebSphere Everyplace® Deployment Server
- ▶ Manage:
 - Tivoli Access Manager
 - Tivoli Federated Identity Manager

Business Process Management scenario

Business Process Management leads to business innovation and optimization by implementing business strategy through modeling, developing, deploying, and managing business processes throughout the entire life cycle. Business Process Management acts as an enabler for businesses in defining and implementing strategic business goals and then measuring and managing a company's financial and operational performance against these goals.

Note: A more detailed description of the Business Process Management scenario can be found in *Patterns: SOA Foundation - Business Process Management Scenario*, SG24-7234.

Business Process Management combines business processes, information, and IT resources, aligning the organization's core assets, people, information, technology, and processes to create a single integrated view, with real-time intelligence, of both its business measurements and IT system performance.

The IBM process integration portfolio provides capabilities required for the delivery of the comprehensive enterprise wide Business Process Management strategies and solution. It offers a holistic approach to transform and manage a business by aligning strategic and operational objectives with business activities and supporting IT services. The IBM Business Process Management solution includes development tools used to implement custom artifacts that leverage the infrastructure capabilities, and business performance management tools, used to monitor and manage the runtime implementations at both the IT and business process levels.

The core IBM products used for the Business Process Management scenario are:

- ▶ Model: WebSphere Business Modeler V6
- ▶ Assemble: WebSphere Integration Developer V6
- ▶ Deploy: WebSphere Process Server V6
- ▶ Manage:
 - WebSphere Business Monitor V6
 - Tivoli Composite Application Manager for SOA V6

The additional IBM products used for this scenario, depending on customer requirements, are as follows:

- ▶ Model: Rational Software Architect V6
- ▶ Deploy:
 - WebSphere Portal
 - WebSphere Adapters
- ▶ Manage:
 - Tivoli Access Manager
 - Tivoli Federated Identity Manager
- ▶ Governance: WebSphere Service Registry and Repository

Information as a Service scenario

The Information as a Service scenario is used to demonstrate unified and trusted information available as services from multiple data sources. This scenario includes content management, e-forms, security, business intelligence, information integration through *just-in-time* integration, and ETL (extract, transform, and load).

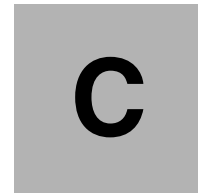
The key drivers for this scenario are to facilitate better decision making and better information sharing between business operations.

The core IBM products used in the Information as a Service scenario are as follows:

- ▶ Model:
 - WebSphere Business Modeler
 - Rational Data Architect
- ▶ Assemble:
 - WebSphere Integration Developer
 - WebSphere DataStage™ Designer
- ▶ Deploy:
 - WebSphere Information Server
 - WebSphere DataStage Integration Suite
- ▶ Manage:
 - WebSphere Business Monitor
 - Tivoli CAM for SOA

Additionally, the following IBM products can be considered by a customer, depending on specific requirements:

- ▶ Deploy:
 - WebSphere Portal
 - WebSphere Adapters
 - Workplace™ Forms View / Server
- ▶ Manage:
 - Tivoli Access Manager
 - Tivoli Federated Identity Manager



Security standards and technology

This appendix provides background information and references for the important standards and technology relevant to security in an SOA.

Web services security specifications

The Web services security model introduces a set of individual interrelated specifications to form a layered approach to security. A motivation and overview of these specifications can be found in a joint security whitepaper from IBM Corporation and Microsoft Corporation on the IBM developerWorks Web site:

<ftp://www.software.ibm.com/software/developer/library/ws-secmap.pdf>

Figure C-1 illustrates these specifications and the layered approach taken in developing these standards.

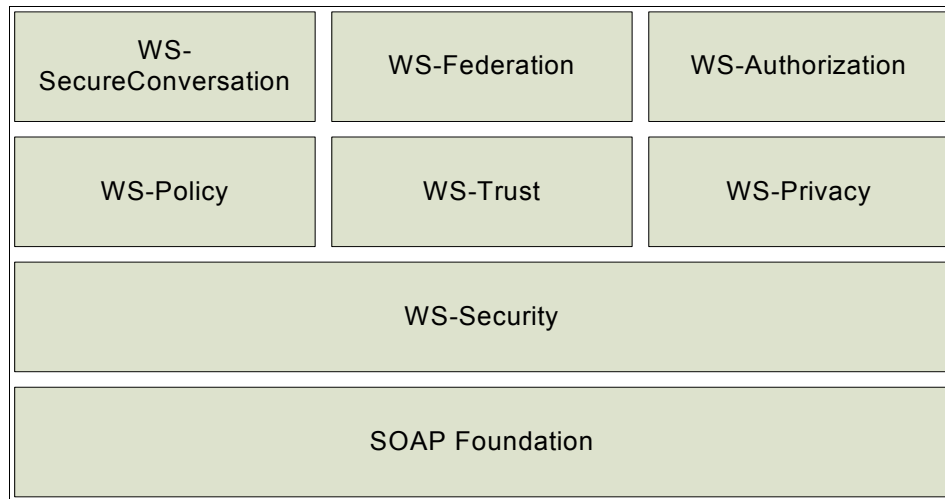


Figure C-1 Web service security specifications

These specifications provide the following capabilities:

- ▶ **WS-Security**
The WS-Security specification provides message-level security which is used when building secure Web services. Message content protection (integrity, confidentiality and authentication) and security token propagation are features of this specification.
- ▶ **WS-Policy**
Describes the capabilities and constraints of the security (and other business) policies on intermediaries and endpoints (for example, required security tokens, supported encryption algorithms, and privacy rules).

- ▶ **WS-Trust**
Describes a framework for trust models that enables Web services to interoperate securely. This specification is responsible for managing trusts and establishing trust relationships.
- ▶ **WS-Privacy**
Describes a model for how Web services and consumers state privacy preferences and organizational privacy practice statements.
- ▶ **WS-Federation**
Describes how to manage and broker the trust relationships in a heterogeneous federated environment, including support for federated identities.
- ▶ **WS-Authorization**
Describes how to manage authorization data and authorization policies.
- ▶ **WS-SecureConversation**
Describes security context exchange and establishing and deriving session keys.

WS-Security

The WS-Security specification provides message-level security. The advantage of using WS-Security instead of SSL is that it can provide end-to-end message level security. This means that the messages are protected even if the message goes through multiple services, or intermediaries. Additionally, WS-Security is independent of the transport layer protocol. It can be used for any SOAP binding, not just for SOAP over HTTP.

As an overview, Figure C-2 shows the elements Web service security elements that may be added to the SOAP header.

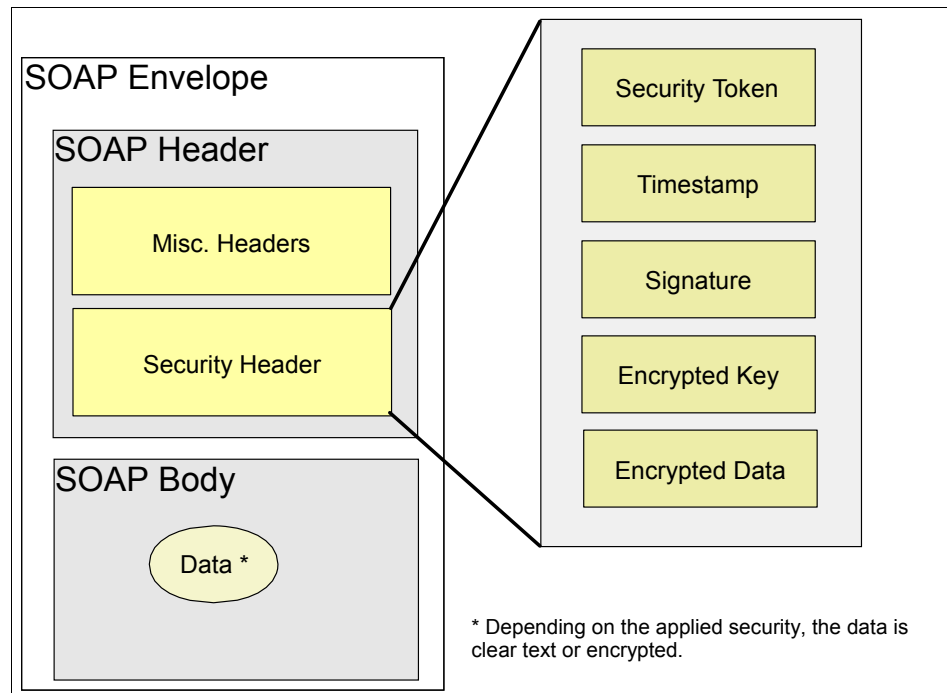


Figure C-2 SOAP message security with WS-Security

The WS-Security specification Version 1.1 was ratified by the OASIS WSS Technical Committee in February 2006. This specification proposes a standard set of SOAP extensions. This specification is flexible and is designed to be used as the basis for securing Web services within a wide variety of security models including PKI, Kerberos, and SSL. It provides support for multiple security token formats, multiple trust domains, multiple signature formats, and multiple encryption technologies to provide integrity or confidentiality.

The WS-Security specification defines the usage of XML Signature and XML Encryption:

- ▶ Message integrity is provided by XML Signature in conjunction with security tokens to ensure that modifications to messages are detected. See: <http://www.w3c.org/Signature>
- ▶ Message confidentiality leverages XML Encryption in conjunction with security tokens to keep portions of a SOAP message confidential. See: <http://www.w3c.org/Encryption>

The specification includes security token propagation, message integrity, and message confidentiality. However, these mechanisms by themselves do not address all the aspects of complete security solution. Therefore, WS-Security represents only one of the layers in a complex secure Web services solution design.

More information about the OASIS Web services Security specifications is available from:

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

WS-Policy

WS-Policy provides a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML Web services-based system. WS-Policy defines a framework and a model for the expression of these properties as policies. Policy expressions allow for both simple declarative assertions as well as more sophisticated conditional assertions.

WS-Policy defines a policy to be a collection of one or more policy assertions. Some assertions specify traditional requirements and capabilities that will ultimately manifest on the wire (for example, authentication scheme and transport protocol selection). Some assertions specify requirements and capabilities that have no wire manifestation yet are critical to proper service selection and usage (for example, privacy policy and QoS characteristics). WS-Policy provides a single policy grammar to allow both kinds of assertions to be reasoned about in a consistent manner.

WS-Policy stops short of specifying how policies are discovered or attached to a Web service. Other specifications are free to define technology-specific mechanisms for associating policy with various entities and resources. Subsequent specifications will provide profiles on WS-Policy usage within other common Web services technologies.

The specification that makes up WS-Policy is available for download from IBM developerWorks at:

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

This is an excerpt from the IBM developerWorks definition of WS-Policy.

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.

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/>

This is an excerpt from the IBM developerWorks definition of WS-Trust.

WS-Federation

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-Federation Passive describes how to leverage the advantages of a federation relationship such as single sign-on in a passive client environment. Because this solution leverages the WS-Security foundation of the infrastructure support, the same components used to provide a passive client solution may be leveraged for an active client solution.

The 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.

WS-SecureConversation

The Web Services Secure Conversation Language (WS-SecureConversation) is built on top of the WS-Security and WS-Policy models to provide secure communication between services. WS-Security focuses on the message authentication model, but not a security context, and thus is subject to several forms of security attacks. This specification defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation.

By using the SOAP extensibility model, modular SOAP-based specifications are designed to be composed with each other to provide a rich messaging environment. As such, WS-SecureConversation by itself does not provide a complete security solution. WS-SecureConversation is a building block that is used in conjunction with other Web service and application-specific protocols (for example, WS-Security) to accommodate a wide variety of security models and technologies.

The specification that makes up WS-SecureConversation is available for download from IBM developerWorks at:

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

This is an excerpt from the IBM developerWorks definition of WS-SecureConversation.

WS-SecurityPolicy

The Web Services Security Policy Language (WS-SecurityPolicy) specification defines a set of security policy assertions that apply to Web Services Security: SOAP Message Security, WS-Trust, and WS-SecureConversation. This specification takes the approach of defining a base set of assertions that describe how messages are to be secured. Flexibility with respect to token types, cryptographic algorithms, and mechanisms used, including using transport-level security, is part of the design and allows for evolution over time. The intent is to provide enough information for compatibility and interoperability to be determined by Web services participants, along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

The specification that makes up WS-SecurityPolicy is available for download from IBM developerWorks at:

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

This is an excerpt from the IBM developerWorks definition of WS-SecurityPolicy.

WS-Provisioning

WS-Provisioning describes the APIs and schema 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.

More information

Because Web services security is a quickly evolving field, it is essential for developers and designers to regularly check for recent updates. In this section, we provide some of the most important entry points for your exploration.

- ▶ The XML Signature Workgroup home page can be found at:
<http://www.w3.org/Signature/>
- ▶ The XML Encryption Workgroup home page can be found at:
<http://www.w3.org/Encryption/>
- ▶ The WS-Security specification 1.0 can be found at:
<http://www.ibm.com/developerworks/library/ws-secure/>
- ▶ The whitepaper of Web services security roadmap can be found at:
<http://www.ibm.com/developerworks/webservices/library/ws-secmap/>
- ▶ The OASIS WS-Security 1.0 and token profiles can be found at:
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

Security Assertion Markup Language

Security Assertions Markup Language (SAML) 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 Security Services Technical Council (SSTC). 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.

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.

More information about the SAML specification is available from:

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

Liberty

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

The *Liberty 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 federation specification is Version 1.2, which was released November 2003.

The Liberty approach is based on business affiliates forming *circles of trust*. The Liberty circles of trust are 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.”

For more information about Liberty Alliance, see:

<http://www.projectliberty.org>

eXtensible Access Control Markup Language

eXtensible Access Control Markup Language (XACML) is an initiative to develop a standard for access control and authorization systems. It describes both, a common language for expressing access control policies to describe general access control requirements and a request/response language that describes how to form a query to determine if a given action is allowed or not and how to interpret the result.

XACML addresses several use cases:

- ▶ Define a policy.
- ▶ Gather required data for policy evaluation.
- ▶ Evaluate policy.

- ▶ Enforce policy.

More information about XACML can be found at the OASIS web site:

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

Java Authorization Contract for Containers

Java Authorization Contract for Containers (JACC) was introduced in the J2EE 1.4 specification to address some problems and limitation of earlier definitions:

- ▶ All access decisions were made by the application server, unless proprietary interfaces were used for third party plug-ins.
- ▶ There were no standards for integration of application servers with authorization service providers. There was no standard representation of application security policy (roles, resources, and resource-to-role mappings) and no standard interface for access decision (declarative or programmatic).

JACC allows third party authorization service providers to plug into application servers like WebSphere using standard interfaces for policy configuration and access decisions. JACC defines new Permission classes to handle both the EJB and the Web permissions required by “security constraints” in J2EE deployment descriptors. A J2EE role is a named collection of these permissions.

Note: JACC does not specify a standard interface for principals (users and groups) to roles mapping.

JACC defines a standard *contract* (interfaces and rules) that allows authorization framework providers to plug into J2EE application containers to provide authorization policy management and access decision services. Figure C-3 shows these relationships.

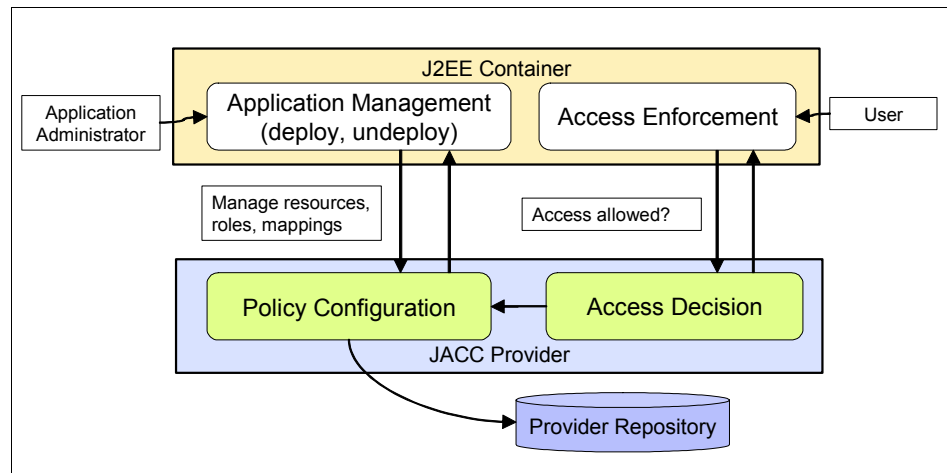


Figure C-3 Java Authorization Contract for Containers (JACC)

More information about JACC can be found at:

<http://java.sun.com/j2ee/javaacc/index.html>

Service Provisioning Markup Language

Service Provisioning Markup Language (SPML) provides an XML framework for managing the provisioning and allocation of identity information and system resources within and between organizations. SPML V2.0 was ratified as an OASIS standard in April 2006.

SPML defines four main conceptual elements:

- ▶ Requesting Authority

A Requesting Authority (RA) constructs and issues well-formed SPML requests and sends them to a Provisioning Service Provider (PSP). An RA may be a portal or other user-facing application, or could be an HR system considered the originating identity repository in an enterprise.

- ▶ Provisioning Service Provider

A Provisioning Service Provider (PSP) is a software system that receives SPML requests and processes them with its internal knowledge of systems

that it manages. A single SPML request may generate multiple provisioning operations within the PSP. An example of a PSP might be a commercial identity provisioning product, such as IBM Tivoli Identity Manager.

- ▶ Provisioning Service Target

A Provisioning Service Target (PST) represents an endpoint that the PSP manages. An example of a PST might be the Tivoli Identity Manager adapter for Active Directory®.

- ▶ Provisioning Service Object

A Provisioning Service Object (PSO) is an object on a PST. An example of a PSO might be an account in RACF. The SPML specification does not restrict itself to only managing user accounts.

The high-level interaction between these components is shown in Figure C-4.

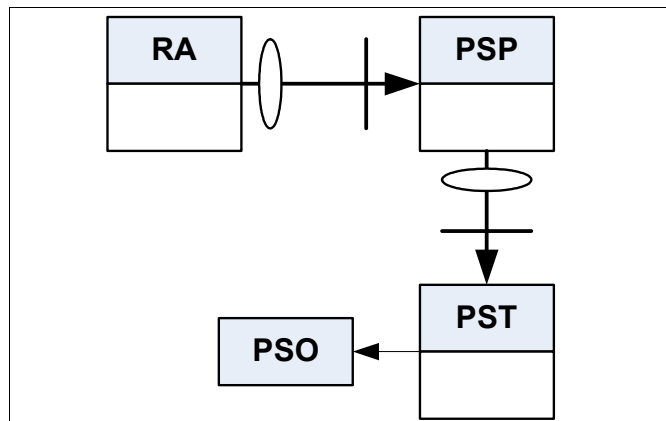


Figure C-4 Conceptual elements in the SPML domain model

Despite the second version of SPML already being ratified as a standard, additional work is required to cater for comprehensive implementations. Examples include:

- ▶ There is no concept of ownership of PSOs.
- ▶ Workflow interactions, such as request approval, cannot be represented in SPML.
- ▶ Protocol bindings to transports are not well-defined at this time. Integration with development tools is limited, and achieving interoperability is then based on interaction contracts beyond the SPML standard.
- ▶ Securing SPML messages is not well-defined.

For more information about SPML and its progress as an evolving standard, consult the OASIS Provisioning Services Technical Committee site:

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

Identity Attribute Service (IdAS)

To support a dynamic environment where sources of identity information may change, it is necessary to provide a common means to access identity and attribute information from across multiple identity repositories. The IdAS virtualizes identity sources and provides a unified view of identity information. For more information, please refer to the project Higgins wiki at:

http://wiki.eclipse.org/index.php/Higgins_Wiki

z/OS Security

z/OS security can be grouped into two major base components:

- ▶ The System Authorization Facility (SAF)
- ▶ RACF

System Authorization Facility

The System Authorization Facility is part of the z/OS operating system and provides the interfaces to the callable services provided to perform authentication, authorization, and audit logging.

SAF does not require any other product as a prerequisite, but overall system security functions are greatly enhanced and complemented if it is used concurrently with RACF. The key element in SAF is the SAF router. This router is always present, even when RACF is not present.

The SAF router provides a common focal point for all products providing resource control. This focal point encourages the use of common control functions shared across products and across systems. The resource managing components and subsystems call the SAF router as part of decision-making functions in their processing, such as access-control checking and authorization-related checking. These functions are called control points.

The system authorization facility (SAF) conditionally directs control to RACF (if RACF is present), or to a user-supplied processing routine, or both, when receiving a request from a resource manager.

Figure C-5 shows the basic functionality of SAF.

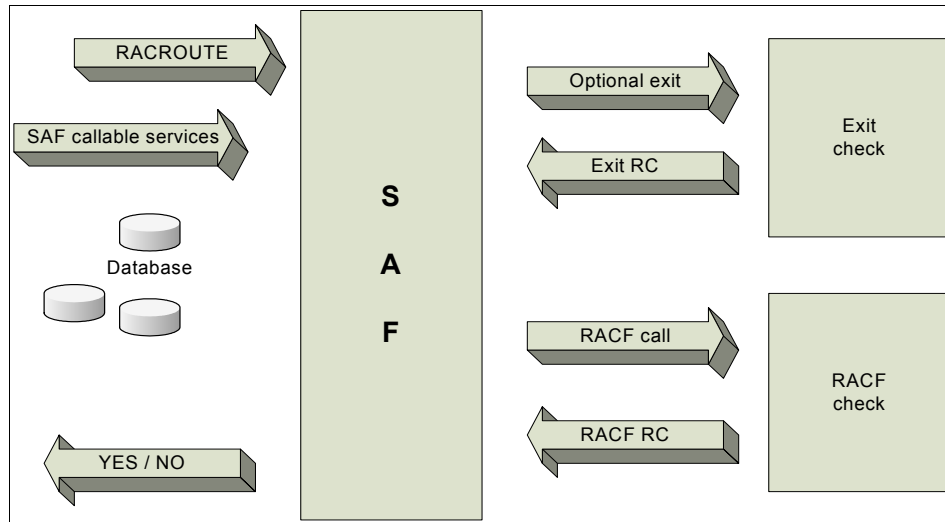


Figure C-5 System Authorization facility

RACF

Resource Access Control Facility (RACF) is an add-on software product that provides security for a mainframe system.

RACF protects resources by granting access only to authorized users of the protected resources. RACF retains information about users, resources, and access authorities in special structures called profiles in its database, and it refers to these profiles when deciding which users should be permitted access to protected system resources.

To accomplish its goals, RACF gives you the ability to:

- ▶ Identify and authenticate users
- ▶ Authorize users to access protected resources
- ▶ Log and report various attempts of unauthorized access to protected resources
- ▶ Control the means of access to resources
- ▶ Allow applications to use the RACF macros

RACF uses a user ID and a system-encrypted password to perform its user identification and verification. The user ID identifies the person to the system as a RACF user. The password verifies the user's identity. Often exits are used to

enforce a password policy such as a minimum length, lack of repeating characters, or adjacent keyboard letters, and also the use of numerics as well as letters.

To get an cursory understanding of how authorization works and to explain the communications involved, an authorization check is performed the following way (see Figure C-5 on page 385):

1. A user requests access to a resource using a resource manager.
2. The resource manager issues a RACROUTE request to see if the user can access the resource.
3. RACF refers to the RACF database or in-storage data.
4. Retrieves data for the profile.
5. Based on the information in the profile, RACF passes the resulting status code for the request to the resource manager.
6. The resource manager grants or denies the request.

RACF does not decide whether the request is granted or not. It returns a status code to the resource manager, and the resource manager makes the decision. RACF will return one of four status codes meaning:

- ▶ The user has the access right.
- ▶ The user does not have access.
- ▶ It is unknown whether the user has the access right or not.
- ▶ RACF is not working.

The resource manager uses this return to make a decision.

An alternative to the RACF password provided by the RACF secured sign-on function is the Passticket. The RACF Passticket is a one-time-only password that is generated by a requesting product or function. It is an alternative to the RACF password that removes the need to send RACF passwords across the network in clear text. It makes it possible to move the authentication of a mainframe application user ID from RACF to another authorized function executing on the host system or to the work station local area network (LAN) environment.

Tip: To learn more about SAF and how it works with RACF, you can look in:

- ▶ OS/390® SecureWay® Security Server RACROUTE Macro Reference

<http://publibz.boulder.ibm.com/epubs/pdf/ich1c610.pdf>

- ▶ OS/390 SecureWay Security Server RACF Callable Services

<http://publibz.boulder.ibm.com/epubs/pdf/ich1d121.pdf>

To learn more about Passticket, please refer to Chapter 11, “The RACF Secured Signon PassTicket”, in *z/OS V1R4.0 Security Server RACF Macros and Interfaces*, SA22-7682, found at:

<http://publibz.boulder.ibm.com/epubs/pdf/ichza330.pdf>



D

Additional material

This IBM Redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this IBM Redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247310>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbook form number, SG247310.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
SG247310.zip	See details below

The zip file contains the following files:

ITSOBanker2006.ear	Bank Web service with no security.
ITSOBankerClient2006.ear	Bank Web service client with no security.
SecureITSOBanker2006.ear	Bank Web service with security added.
SecureITSOBankerClient2006.ear	Bank Web service client with security added.
client.jks	Key store for the client application.
was.jks	Key store for the Web service application.
soa-jaas-login.jar	JAAS login module that calls Federated Identity Manager Trust service.
wssm_one-to-one.xsl	Identity mapping rule for Federated Identity Manager WSSM partner chain.
com.tivoli.am.fim.ldap.plugin_1.0.zip	Trust service module for identity mapping using LDAP lookups.
bank2005.zip	This file contains the CICS portion of the scenario. See more information below.

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this subdirectory (folder). Chapter 8, “Technical implementation” on page 193 provides detailed instructions on how to use these files.

Installing the CICS portion of the ITSOBank scenario

Perform the following tasks:

1. Unzip the supplied bank2005.zip to a temporary directory.
2. On the z/OS side, create a partitioned dataset library with the FB80 format for the source files.
3. Upload all the files from the bank2005.zip as members to this datasets. You can ignore the extensions. The members would be:

DPLCSHOW	Assembler source for 3270 transaction
DPLCMAP	CICS map source
DPLC	Main C program source
ACCTBNTB	SPUFI source for account table

EMPLOYEE SPUFI source for employee data

CSDUP CICS resource definition source

4. Compile the program and map the source into a load library that is concatenated to the CICS's DFHRPL.
 - DPLCMAP can be compiled using the procedure DFHMAPS or DFHMAPT.
 - DPLCSHOW can be assembled and link-edited using the procedure DFHEITAL.
 - DPLC must be compiled through the DB2 pre-processor and the CICS translator. You can use the supplied DB2 sample DSNHC procedure and invoke the CICS procedure DFHYITDL.
5. Create DB2 tables. You may need to work with a DB2 administrator for performing this task. The sample SPUFI files will create the tables on the public database DSNDB04 using the default storage group. Run SPUFI using the EMPLOYEE source and then the ACCTBNTB. All return codes should be 0000.
6. Create a CICS resource definition using the source CSDUP. This creates a group called BANK2005. You should include this group into the startup list for the CICS initialization to be automatically installed when you start CICS. You can interactively install the group using the CEDA INSTALL GROUP(BANK2005) command from a CICS session.

7. Test the transaction by invoking DPLS. You should receive miscellaneous data as shown in Figure D-1.

```
DPLC
      Return Data from Program DPLC
Field 1: 0000
Field 2: .00
Field 3:      Date:08/
Field 4: 02/2007 Time:1
Field 5: 0:12:50 Applid:S
Field 6: CSCPA2B Sys
Field 7: id:PA2B Userid:C
Field 8: ICSUSER Start
Field 9: code:D Tran: DPL
```

Figure D-1 CICS DPLC output

8. To prepare the connection from the CICS Transaction Gateway, you must define the necessary resources for the CICS Transaction Gateway to access this CICS. Additional definitions must include CONNECTION and SESSION resources.
9. Prepare WebSphere Application Server for CICS connection. Define a connection resource to access CICS using the appropriate cicsecl.rar file from the CICS Transaction Gateway version that you use. The rar file can be installed using the WebSphere administrative console:
 - a. Invoke **Resources** → **Resource adapters** → **Resource adapters** and click **Install RAR**.
 - b. Select the cicsecl.rar for the appropriate node and use the default options.
 - c. Save the configuration to the master configuration.
 - d. Invoke **Resources** → **Resource adapters** → **J2C Connection Factories** and click **New**.
 - e. Define new J2C connection factories that identifies the CICS that you are connecting, such as SCSCPA2B (see Figure D-2 on page 393). We recommend you use the APPLID name of the CICS region, as this would be the same name to use in CICS transaction gateway connection.

[J2C connection factories](#) > **New**

Use this page to create a connection factory for use with the resource adapter. The connection factory is a collection of configuration values that define a WebSphere(R) Application Server connection to your Enterprise Information System (EIS). The connection pool manager uses these properties as directions for allocating connections during runtime. You can configure multiple connection factories for each resource adapter.

Configuration

General Properties

* Scope

Provider

* Name

JNDI name

Description

The additional properties will not be available until the general properties for this item are applied or saved.

Additional Properties

- Connection pool properties
- Advanced connection factory properties
- Custom properties

Related Items

Figure D-2 WebSphere configuration

- f. Click **Apply**. The additional properties will then be enabled.
- g. Go to Custom properties in the Additional properties.

h. Set the custom properties similar to Figure D-3; where:

- ConnectionURL** This reflects the target TCP/IP address or host name of the z/OS machine on which the CICS transaction gateway is running.
- PortNumber** This is the listening port number for CICS transaction gateway; the default value is 2006.
- ServerName** This is the APPLID of the CICS region that you want to connect to.
- TPNName** This is the CICS transaction name that you would use. In the BANK2005, it is called DPLC.

[J2C connection factories](#) > [SCSCPA2B](#) > Custom properties

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

☰ Preferences

Name	Value	Description	Required
TraceLevel	1	TraceLevel	false
TPNName	DPLC	TPNName	false
Password		Password	false
UserName		UserName	false
TranName		TranName	false
ConnectionURL	tcp://wtsc58.itso.ibm.com	ConnectionURL	false
ServerName	SCSCPA2B	ServerName	false
ClientSecurity		ClientSecurity	false
KeyRingPassword		KeyRingPassword	false
SocketConnectTimeout	0	SocketConnectTimeout	false
PortNumber	2006	PortNumber	false
KeyRingClass		KeyRingClass	false
ServerSecurity		ServerSecurity	false
Total 13			

Figure D-3 J2C connection factory custom properties

10. Update the resource mapping for the enterprise application to use the newly created J2C connection factory to connect to CICS. This is usually found under **Applications** → **Enterprise Applications**, and select the application that you are using. Go to the Resource references link and make sure that the content of the Target Resource JNDI Name field contains the J2C connection factory that you define in step 9 on page 392.

11. Save the WebSphere configuration to master configuration and try the application.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this IBM Redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 398. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Develop and Deploy a Secure Portal Solution Using WebSphere Portal V5 and Tivoli Access Manager V5.1*, SG24-6325
- ▶ *Develop and Deploy a Secure Portal Solution Using WebSphere Portal V5 and Tivoli Access Manager V5.1*, SG24-6325
- ▶ *Enterprise Business Portals II with IBM Tivoli Access Manager*, SG24-6885
- ▶ *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014
- ▶ *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394
- ▶ *IBM Tivoli Composite Application Manager V6.0 Family: Installation, Configuration, and Basic Usage*, SG24-7151
- ▶ *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316
- ▶ *Patterns: SOA Foundation - Business Process Management Scenario*, SG24-7234
- ▶ *Patterns: SOA Foundation Service Connectivity Scenario*, SG24-7228
- ▶ *Understanding SOA Security Design and Implementation*, SG24-7310
- ▶ *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- ▶ *WebSphere Version 6 Web Services Handbook Development and Deployment*, SG24-6461

Other publications

These publications are also relevant as further information sources:

- ▶ Lansiti, et al., *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, Harvard Business School Publishing, Press, 2004, ISBN 1591393078
- ▶ “Daimler’s New Way to Make Cars: Let Someone Else Do It”, in *Forbes*, August 16, 2004

Online resources

These Web sites are also relevant as further information sources:

- ▶ The Web services security model introduces a set of individual interrelated specifications to form a layered approach to security. An overview of these specifications can be found on the IBM developerWorks site:

<ftp://www6.software.ibm.com/software/developer/library/ws-secmap.pdf>

- ▶ Many open standards in the areas of security, SOA, Web Services, and similar topics are documented at the Organization for the Advancement of Structured Information Standards (OASIS):

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

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, IBM 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 policy 145
 - management policy 86
 - policy management 46
- Access Manager 110, 363
 - Common Auditing and Reporting Service 307
 - protected object space 274
 - transport level security 181
 - trust service authentication 195
 - trust service authorization 195
 - Web service authorization 274
- Access Manager for e-business
 - audit 185
 - authorization audit 179
 - JACC provider 178
 - policy decision and enforcement 188
 - Web service authentication 170
- Access Manager for Operating Systems
 - Data Protection and Disclosure Control 190
 - protection for data at rest 180
- Access Services 22, 352
- account
 - re-certification 161
- accountability
 - ... for identity propagation 52
 - ... for policy changes 43
- addressing 90
- application
 - chain 259
 - client 198
 - configuration 202
 - response consumption 214
 - trust chain 266
- approach to creating service
 - bottom-up 342
 - top-down 342
- assemble 14, 348
- audit 11, 28, 82, 161
 - authorization decisions 179
 - data protection 137
 - policy 109
 - requirements 43

- Audit Services 36, 76, 105, 136, 183
- authentication 26, 142, 160
 - Web service 170
- Authentication Services 32, 71, 99, 131
 - externalization 131
- authorization 27, 160
 - policy 33, 109
 - role based 73
 - Web service 170
- Authorization and Privacy Services 73, 102, 134
- Authorization Services 33
- availability 28
 - compliance 188

B

- Basel II 112
- basic authentication 309
- biometric authentication 32
- bottom-up 342
- boundary service 9
- business
 - compliance 44
 - context 3
 - policy 108
 - requirement 78, 94
 - requirements 158, 327
 - trust management 45
- Business Application Services 22, 352
- Business Innovation and Optimization Services 23
- Business Innovation Services 353
- business process
 - decomposition 4
 - modeling 14
 - security 25
- Business Process and Policy Management 48, 87, 115, 147
- Business Process Execution Language for Web Services 343
- Business Process Management scenario 367
- Business Security Services 28, 43, 66, 82, 111, 143, 189

C

- call back handler 175, 205
- CBE 36
- certificate
 - key store 201
- CICS 357–358
 - audit 185
 - Connection Factory 248
 - JCA adapter 195
 - resource adapter 245
 - Web service exposure 155
 - Web service transactions 161
- CICS Transaction Gateway 155, 358
 - authentication 160, 177
 - JAAS login 195
 - JCA connector
 - trace 302
 - security token exchange 173
 - transport level security 181
- COMMAREA 358
- Common Auditing and Reporting Service 307
- Common Base Event 307
 - see CBE
- compliance 8, 11, 15, 28, 43–44, 76, 82–83, 105, 160, 188
 - objective 143
 - reporting 112
 - reports 136
- component based application 329
- componentization 5
- composite application 10, 19, 325, 356
- confidentiality 27, 43, 160, 215, 226
 - ... of messages 35, 180
- Confidentiality and Integrity Services 75, 104, 135, 180
- Confidentiality Services 35
- consent 87
- consume Web service 342
- consumer 19, 22, 24
 - portal 51
- corporate policy 43
- CTG
 - See CICS Transaction Gateway

D

- data
 - ... at rest 27, 35, 180
 - ... in process 27

- ... in transit 27, 180
- classification 135
- delivery 36
- encryption 27
- format translation 6
- origin 36
- protection 11, 75, 104
 - management 47
- Data Protection and Disclosure Control 49, 86, 113, 145, 190
- Data Protection Services 35, 180
- DataPower 353
 - appliance 165, 170
 - audit 185
 - authenticating and authorizing external requests 173
 - message level security 183
 - policy decision and enforcement 187
 - Toolkit 363
 - transport level security 180
 - XS40 363
- declarative security 50, 80
 - constraints 196
 - enforcement 34
- decomposition
 - ... of business processes 4
- delivery of data 37
- deploy 15, 348
- deployment
 - architecture 53
 - descriptor 34, 82, 202
 - confidentiality 215
 - integrity 215
 - WS Extension 202
- Development Services 23, 354
- digital certificate
 - message level security 182
- digital signature 27, 35, 37
- Direct Exposure Architectural Pattern 62
- disclosure control 47, 87, 113, 146
- discovery 326
- dynamic invocation 342
- dynamic proxy 342

E

- Eclipse 332
- EDI
 - security 37

EIS
 See Enterprise Information System
EJB container 342
Electronic Data Interchange
 see EDI
encryption 27, 35
Enterprise Architect 350
Enterprise Information System 357
Enterprise Service Bus
 see ESB
enterprise transformation 357
ESB 22, 50, 90, 353
 mediation module 93, 98
 message protection policies 108
 policy enforcement 111
expose J2EE artifact as service
 servlet 342
 stateless session EJB 342
expose Web services 341
eXtensible Access Control Markup Language
 see XACML
Extensible Markup Language
 See XML

F

Federated Identity Manager
 audit 185
 audit events 313
 call back handler 175
 Common Auditing and Reporting Service 307
 mapping module 269
 policy decision and enforcement 187
 SAML assertion 234
 Security Token Service 170
 token
 generator 175
 token consumer 175, 236
 transport level security 181
 trust chain 282
 trust chain editor 269
 trust service 253, 259
 trust service APIs 269
 Web Service Security Management 259
 Web Services Security Management 175
federated provisioning 32, 69, 97, 126, 136
federated single sign-on 120, 126, 130, 133
federation 52, 70
 ... of identities 169

 policy 85
flexible architecture 328
flow control 6
functional requirements 161

G

Gateway 362
getting started 333, 335
global security 169
Governance
 See SOA Governance
governance 12
 activities 143
 framework 83
Governance, Risk and Compliance 44, 48, 83, 112, 143

H

Health Insurance Portability and Accountability Act 112
high availability 28
HIPAA 112
HR identity feed 46

I

IBM CICS Transaction Gateway
 see CICS Transaction Gateway
IBM CICS Transaction Server
 see CICS
IBM Rational Software Architect
 see Rational Software Architect
IBM Service Integration Maturity Model 335
IBM SOA Foundation 338
 scenarios 49
IBM SOA Reference Architecture 21
IBM SOA Security Reference Model 28, 66, 94, 122, 164
IBM Tivoli Access Manager
 see Access Manager
IBM Tivoli Access Manager for e-business
 see Access Manager for e-business
IBM Tivoli Access Manager for Operating Systems
 see Access Manager for Operating Systems
IBM Tivoli Composite Application Manager for SOA 363–365
IBM Tivoli Directory Server
 see Tivoli Directory Server

- IBM Tivoli Federated Identity Manager
 - see Federated Identity Manager
- IBM Tivoli Identity Manager
 - see Identity Manager
- identity 25
 - assertion 72
 - federation 52, 70, 98, 126, 169
 - feed 46, 113
 - foundation 68, 96, 124, 167
 - management 84, 125, 144
 - mapping 26, 52, 98, 160, 169
 - mapping rule 263
 - propagation 6, 9, 54
 - provider 126
 - provisioning 31, 69, 97, 125, 136, 144, 168
 - synchronization 124
 - token 26, 32
- Identity and Access 46, 48, 84, 113, 144
- Identity Manager
 - audit 185
 - policy decision and enforcement 188
 - transport level security 181
 - user provisioning 168
- Identity Services 31, 124
- IMS 357
- Indirect Exposure Architectural Pattern 65
- Information as a Service scenario 368
- Information Services 22, 352
- Infrastructure Services 23, 354
- integrity 27, 43, 83, 160, 215, 225
 - ... of messages 35
- Integrity Services 35
- Interaction and Collaboration Services scenario 367
- Interaction Services 21, 352
- invoking Web services 342
- IT Security Services 28, 30, 66–67, 95, 124, 166
- IT service management 354
- IT system life cycle 12

J

- J2EE 332, 357
 - audit 186
 - authorization 73
 - deployment descriptor 34
 - role based authorization 178
- JAAS
 - login 177

- login module 175, 195, 253, 278
 - trace 302
- login value 216
- Principal 176
- JACC
 - overview 381
 - provider 178
- Java Authentication and Authorization
 - see JAAS
- Java Authorization Contract for Containers
 - see JACC
- Java Connector Architecture
 - <\$npage 245
- JAX-RPC 342
- JCA
 - adapter 155, 195
 - resource adapter 245

K

- Kerberos 32
- key locator 228, 240
 - Web service key locator 218
- key store 201
 - password 201

L

- LDAP
 - mapping module 278
- liability 84, 112
- Liberty
 - overview 380
- life cycle 12, 347
 - management 46
 - security 14
- Local integration 363
- location transparency 23
- logging 36, 76, 105
- logical architecture 49, 153

M

- manage 12, 15
- mapping
 - ... of identity 26
 - module 269, 278
 - rule 263
- masquerading 35, 180
- mediation module 93, 98

- message
 - authentication 35
 - code 27
 - broker 8
 - confidentiality 35, 160, 180, 182, 375
 - integrity 35, 160, 182, 375
 - protection 75, 82, 139, 171
 - policy 40, 79, 108, 112, 139, 186
 - service 35, 104, 135
 - queuing 7
 - transport level security 180
- Message Brokers Toolkit 366
- message level
 - protection 36, 76, 104
 - security 80, 182, 373
- messaging
 - style 90
 - styles 23
- Methodology 333
- model 14, 347
- model-driven development 333
- monitoring 12
- Monitoring and Reporting 43, 82, 111, 143, 188
- monolithic business application 329
- mutually authenticated SSL 180

N

- nonce 35
- non-disclosure 35
- non-functional requirements 162
- Non-repudiation Services 36, 77, 106, 137

O

- Optimization Services 353
- origin of data 37

P

- part reference 221
- partner chain 259
- Partner Services 22, 352
- passticket 71, 79, 86, 98, 128, 136, 177, 195, 253, 278
- password
 - policy 85, 109, 145
 - enforcement 142
 - synchronization 69, 113
 - weakness 85

- Patterns 333
- performance
 - compliance 188
- Personally Identifiable Information 27, 33, 74
- pleadable security 158
- point to point protection 76, 104
- policy
 - based provisioning 69
 - management 10, 38, 108, 160
 - message protection 79, 108, 139
 - privacy 87, 146
 - provider 79, 109, 140
 - provisioning 97
- Policy Administration 40, 79, 108, 139, 186
- Policy Decision and Enforcement 42, 81, 110, 141
 - scenario
 - Policy Decision and Enforcement 187
- Policy Decision Point 33, 42, 79, 81, 110, 141, 187
- Policy Distribution and Transformation 40, 80, 109, 140
- Policy Enforcement Point 33, 42, 79, 111, 141, 187
 - externalization 131
- Policy Management 48
- portal 51
- privacy 27, 47
 - consent 87, 114
 - policy 33, 87, 146
- Privacy Services 33
- Process modeling 333
- Process Services 22, 352
- programmatic security enforcement 34
- programming model 34
- propagation of identity 9
- protocol
 - transformation 23
 - translation 6
- provider policy 40, 79, 109, 140, 187
- provisioning 31, 69, 125, 136, 144
 - ... of identities 97
 - federation 69, 136
 - policy 31, 85, 97
 - rule 113
 - workflow 85

R

- RACF
 - authentication 160
 - authorization 74

- identity mapping 160
- JAAS login module 253
- overview 385
- Passticket 71, 79, 86, 98, 128, 136, 177, 195, 253, 278
- policy decision and enforcement 188
- Rational Software Architect
 - adding declarative security constraints 196
 - recompilation 199
- re-authentication 132
- re-certification 161
- Redbooks Web site 398
 - Contact us xiv
- Reference architecture 333
- replay attack 263
- reporting 43
- request based provisioning 31
- request consumption 224
- Resource Access Control Facility
 - see RACF
- Resource Adapter Module 245
- response
 - consumption 214
 - generation 237
- reuse 8
- risk
 - management 44, 83, 112, 143, 189
 - password weakness 85
- role based access control 9
- role based authorization 73, 178
- routing 90

S

- SAF
 - see System Authorization Facility
- SAML
 - assertion 32, 85, 136, 161, 195
 - identity token 26
 - overview 379
 - token 79
 - v 1.1 security token 173
 - v 2.0 security token 173, 175–176
- SAML 2.0
 - assertion 234, 268
 - trace 301
 - token creation 272
- Sarbanes-Oxley Act 112
- SCA

- See Service Component Architecture
- scenario
 - application chain 259
 - application client
 - configuration 202
 - Audit Services 183
 - business requirements 158
 - Business Security Services 189
 - CICS Connection Factory 248
 - compliance 188
 - confidentiality 226
 - Confidentiality and Integrity Services 180
 - Data Protection and Disclosure Control 190
 - Data Protection Services 180
 - deploying the application 244
 - functional requirements 161
 - identity
 - federation 169
 - foundation 167
 - provisioning 168
 - integrity 225
 - IT infrastructure 154
 - IT Security Services 166
 - ITSOBanker2006.ear 198
 - ITSOBankerClient2006.ear 198
 - JAAS login module 253, 278
 - key locator 228
 - logical architecture 153
 - message protection 171
 - policies 186
 - Monitoring and Reporting 188
 - non-functional requirements 162
 - partner chain 259
 - Policy Decision Points 187
 - Policy Enforcement Point 187
 - process flow 194
 - provider policies 187
 - request consumption 224
 - response generation 237
 - risk management 189
 - SAML 2.0 assertion 268
 - Security Policy Infrastructure 186
 - security token exchange 171
 - service provider token consumer 259
 - SOA Governance Board 189
 - solution architecture 164
 - technical implementation 193
 - technical requirements 159
 - token consumer 227

- transport level security 180
- trust anchor 215
- trust chain 282
 - configuration 259
- Trust Management 189
- trust service chains 171
- Web service authorization 274
- Web service flow 194
- SDO
 - See Service Data Objects
- secure sockets layer
 - see SSL
- Secure Systems and Networks 49, 87, 114, 146
- Secure Token Service 169
- security
 - administrator tasks 196
 - compliance 188
 - management 83
 - enforcement 34
 - infrastructure integration challenge 54
 - management 8
 - challenge 8, 54
 - policy 14, 24, 78, 82, 125
 - policy management 28
 - role references 341
 - services
 - standards 37
 - token 26, 32, 52, 70–71, 79, 82, 98, 110, 112, 132, 161, 203
 - exchange 171
 - profile 172
 - propagation 375
 - SAML v 1.1 173
 - SAML v 2.0 173, 175–176
 - trace 298
- Security Assertion Markup Language
 - see SAML
- Security Policy Infrastructure 28, 38, 66, 108, 139, 186
- SecurityTokenRequest 195
- SecurityTokenResponse 195
- SEI
 - See service endpoint interface
- self-care 52
- self-service 69
- separation of duties 34
- service 18, 324
 - component 155
 - consumer 19, 22, 24, 326
 - security policy enforcement 42
 - endpoint interface 341
 - implementation bean 341
 - infrastructure 6
 - interface definition 90
 - management 23
 - managing 12
 - messaging model 90
 - monitoring 12
 - orientation 4, 324
 - life cycle 12
 - reuse 8
 - provider 326
 - security policy enforcement 42
 - token consumer 259
 - registry 20, 326
 - requestor 19
 - security 25
 - specification 90
 - substitution 23
- Service Aggregation scenario 51
- Service Component Architecture 332
- Service Connectivity scenario 50, 360
 - realizations
 - Gateway 362
 - Local integration 363
- Service Creation scenario 49, 357
 - realizations 357
 - Consume services 359
 - Create Web service from WSDL 359
 - Direct exposure 357
 - Indirect exposure 358
- Service Data Objects 332
- Service Level Agreements 356
- service level authorization 73, 102
- Service Provisioning Markup Language
 - see SPML
- Service Registry and Repository
 - see SRR
- Service-oriented Architecture
 - See SOA
- session management 142
 - services 132
- shift in IT driven by business 331
- signature 35, 37
 - validation 218
- signing of Web services 203
- SIMM 335
- Simple Object Access Protocol

- See SOAP
- single sign-off 376
- single sign-on 52, 119, 376
- SLA
 - See Service Level Agreement
- SOA 323–324
 - ... based application 330
 - Adoption 332, 335
 - applications 17
 - Assessment Tool 335
 - business requirements 327
 - challenges 326
 - compliance management 44
 - components
 - service consumer 326
 - service provider 326
 - service registry 326
 - defined
 - by role 324
 - composite application 325
 - service 324
 - service orientation 324
 - Design 355
 - drivers 327
 - achieve better IT use and ROI 328
 - need for flexible architecture 328
 - reduce cycle time and costs 328
 - simplify integration across the enterprise 328
 - support an agile business model 327
 - example approach 333
 - getting started 335
 - IBM SOA Entry Points 336
 - IBM SOA Foundation 338
 - SOA Adoption 335
 - Governance 44, 332, 349, 356
 - overview 18
 - Reference Architecture 21
 - risk management 44
 - security 24
 - audit 28
 - authentication 26
 - authorization 27
 - availability 28
 - compliance 28
 - confidentiality 27
 - deployment architecture 53
 - governance 44
 - identity 25
 - integrity 27
 - logical architecture 49
 - policy
 - management 28
 - privacy 27
 - reference model 28, 66, 94, 122, 164
 - Web services 343
 - why now
 - best practices 332
 - open standards and platforms 332
 - shift in IT driven by business 330
 - SOA enables flexibility 331
- SOA Entry Points 336
 - Connectivity 337
 - Information 337
 - People 336
 - Process 337
 - Reuse 337
- SOA Foundation
 - architecture layers 356
 - life cycle 346
 - Assemble 348
 - Deploy 348
 - Manage 349
 - Model 347
- Reference Architecture 350
 - Access services 352
 - Business application services 352
 - Business innovation services 353
 - Development services 354
 - Enterprise Service Bus 353
 - Information Services 352
 - Interaction services 352
 - Middleware Services view 351
 - Optimization services 353
 - Partner services 352
 - Process services 352
 - Solution view 350
- scenarios 49, 354
 - Business Process Management 367
 - Information as a Service 368
 - Interaction and Collaboration Services 367
 - Service Aggregation 51, 117
 - Service Connectivity 50, 89, 360
 - Service Creation 49, 61, 357
- SOAP 90, 339
 - binding 373
 - body 339
 - encoding rules 339

- envelope 339
- handler 195
- headers 339
- message
 - details 181
 - format 339
 - security 374
- policy assertions 41
- RPC representation 339
- security token 298
- transports 339
- solution architecture 164
- SPML 69, 97
 - overview 382
- SRR 38
- SSL 36, 76, 104, 135, 180
 - mutually authenticated 180
- SSO
 - see single sign-on
- standards
 - security services 37
- static stub 342
- strong authentication 132
- submission of data 37
- synchronization
 - identity 124
 - passwords 69
 - user repository 96
- System Authorization Facility 158, 384
- systems management 8

T

- technical implementation 193
- technical requirements 159
- technology
 - trust management 45
- Tivoli Common Auditing and Reporting Service 184
- Tivoli Directory Server
 - transport level security 181
- token
 - based authentication 32
 - consumer 175, 227
 - generator 175
 - mediation 169
- top-down 342
- transaction
 - auditing 11
- transform 222

- transport level
 - protection 36
 - security 160, 180
- transport of data 37
- transport protocol 90
- trust
 - anchor 215, 226
 - chain 266, 282
 - configuration 259
 - editor 269
 - policy 108
 - relationship 9, 50
- Trust Management 44, 48, 84, 112, 144, 189
- trust service 70–71, 82, 86, 110, 128, 132, 259
 - APIs 269
 - auditing 314
 - authentication 195
 - authorization 195
 - chain 170–171
 - security token exchange 98
 - SecurityTokenRequest 195

U

- UDDI 340
- unauthorized modification 35
- Universal Description, Discovery, and Integration
 - See UDDI
- user
 - consent 87, 114
 - management change 43
 - registry
 - synchronization 68
 - repository 31, 96
 - synchronization 96
 - self-care 113
 - self-service 69
- username token 32

W

- Web service 332, 338
 - ... and SOA 343
 - application
 - client 198
 - trust chain 266
 - authentication 170
 - authorization 170, 274
 - confidentiality 226
 - core elements

- SOAP 339
 - UDDI 340
 - WSDL 339
 - XML 339
 - deployment 255
 - flow 194
 - gateway 165
 - integrity 225
 - J2EE 341
 - JAAS login module
 - trace 302
 - key locator 228, 240
 - message parts dialect 206
 - part reference 221
 - request consumption 224
 - request generation 202
 - response consumption 214
 - response generation 237
 - SAML 2.0 assertion 268
 - security constraints 202
 - security specifications 372
 - signing 203
 - standards 340, 372
 - technologies 338
 - token consumer 227
 - transform 222
 - trust anchor 226
 - WS-Federation 376
 - WS-Policy 375
 - WS-Provisioning 378
 - WS-Security 373
 - WS-SecurityPolicy 378
 - WS-Trust 376
 - Web Service Security Management 259
 - Web Services Description Language
 - See WSDL
 - Web Services for Remote Portlets 122
 - Web services Interoperability Organization 340
 - Web Services Security Management 175
 - WebSphere Application Server
 - audit 185
 - global security 169
 - JACC
 - provider 178
 - message level security 183
 - policy decision and enforcement 188
 - Resource Adapter Module 245
 - security token exchange 172
 - transport level security 180
 - WebSphere Enterprise Service Bus 353, 364
 - WebSphere Integration Developer 364–365
 - WebSphere Message Broker 353
 - workflow
 - provisioning 85
 - WS Extension 202
 - WS-BPEL
 - See Business Process Execution Language for Web Services
 - WSDL 90, 339, 343
 - definition 341
 - wsdl2tfim.sh 266
 - WSEE 341
 - WS-Federation 376
 - WS-Policy 40, 375
 - WS-Provisioning 97, 378
 - WS-Security 27, 76, 135, 255, 373
 - authentication 175
 - message level security 182
 - message protection 104
 - Policy 40
 - Roadmap 373
 - token processing 259
 - Web site 379
 - WS-SecurityPolicy 378
 - WS-Trust 169, 376
- X**
- XACML 40–41
 - overview 380
 - XML 90, 339
 - encryption 374
 - Web site 379
 - firewall 165
 - signature 374
 - Web site 379
 - stylesheets 263
 - XSLT
 - mapping rule 263
 - transformation mapping 269
- Z**
- z/OS
 - RACF 385
 - security 384
 - System Authorization Facility 384



Redbooks

Understanding SOA Security Design and Implementation

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Understanding SOA Security

Design and Implementation

Introducing an SOA security reference architecture

Implementing scenarios based on the IBM SOA Foundation

Deploying SOA using IBM Tivoli security solutions

Securing access to information is important to any business. Security becomes even more critical for implementations structured according to service-oriented architecture (SOA) principles, due to loose coupling of services and applications, and their possible operations across trust boundaries. To enable a business so that its processes and applications are flexible, you must start by expecting changes in both to process and application logic, as well as to the policies associated with them. Merely securing the perimeter is not sufficient for a flexible on demand business.

In this IBM Redbook, security is factored into the SOA life cycle reflecting the fact that security is a business requirement, and not just a technology attribute. We discuss a SOA security model that captures the essence of security services and securing services. These approaches to SOA security are discussed in the context of some scenarios, and observed patterns. We also discuss a reference model to address the requirements, patterns of deployment, and usage, and an approach to an integrated security management for SOA.

This IBM Redbook is a valuable resource to senior security officers, architects, and security administrators.

**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