Tivoli software

IBM

Enterprise Security Architecture

Using IBM Tivoli Security Solutions

- Audit and compliance, access control, identity management, and integrity
- Extensive product architecture and component introduction
- Complete coverage of Tivoli Security solutions

Axel Buecker
Ana Veronica Carreno
Norman Field
Christopher Hockings
Daniel Kawer
Sujit Mohanty
Guilherme Monteiro

Redbooks

ibm.com/redbooks

International Technical Support Organization

Enterprise Security Architecture Using IBM Tivoli Security Solutions

August 2007

SG24-6014-04

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page iii.

Fifth Edition (August 2007)

This edition applies to the following IBM Tivoli products: Access Manager for e-business 6.0, Access Manager for Business Integration 5.1, Access Manager for Operating Systems 6.0, Access Manager for Enterprise Single Sign-On Version 6.0, Directory Server 6.0, Directory Integrator 6.1.1, Identity Manager 4.6, Identity Manager Express Version 4.6, Federated Identity Manager 6.1, Federated Identity Manager Business Gateway 6.1, Security Operations Manager 3.1, and Security Compliance Manager 5.1.1. Various other related IBM, Tivoli and Lotus products are mentioned in this book.

Notices

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

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

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

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

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

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

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

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

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

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

Trademarks

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

Redbooks (logo) ®
developerWorks®
e-business on demand®
eServer™
iSeries®
pSeries®
z/OS®
zSeries®
AIX 5L™
AIX®
AS/400®
Cloudscape™
CICS®
DataPower®
Domino®
DB2 Universal Database™
DB2®
Everyplace®
HACMP™
Informix®
Internet Scanner®
IBM®
Lotus Notes®
Lotus®
MQSeries®
Notes®
OS/390®
OS/400®
Proventia®
Redbooks®
RACF®
RDN™
Sametime®
SecureWay®
System i™
System p™
System x™
System z™
Tivoli Enterprise™
Tivoli Enterprise Console®
Tivoli®
Update Connector™
WebSphere®

The following terms are trademarks of other companies:

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

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

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

PostScript, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Enterprise JavaBeans, EJB, Java, Java Naming and Directory Interface, JavaBeans, JavaScript, JavaServer, JDBC, JMX, JSP, JVM, J2EE, Solaris, Sun Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Access, Active Directory, ActiveX, Expression, Internet Explorer, Microsoft, Visual Basic, Visual Studio, Windows NT, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

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

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

Contents

Preface

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

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

Part 1, “Terminology and infrastructure” on page 1, introduces the foundation needed for an enterprise-wide security architecture. We discuss the business drivers, foundation IT technologies and the network topologies you will encounter when designing your infrastructure. Finally, two specific components will be explained in more detail because they belong in every IT infrastructure today: the LDAP-based IBM Tivoli Directory Server and the IBM Tivoli Directory Integrator.

Part 2, “Managing access control” on page 161, focuses on the access control systems of the security architecture and introduces the IBM Tivoli Access Manager components.

Part 3, “Managing identities and credentials” on page 507, takes a closer look at the identity and credential systems with the IBM Tivoli offerings Identity Manager and Directory Integrator.

Part 4, “Managing federations” on page 653, takes us into the rapidly expanding world of federated identity management and Web services security and provisioning by introducing the IBM Tivoli Federated Identity Manager.

Part 5, “Managing security audit and compliance” on page 843, examines the audit systems and explains how the IBM Tivoli Security Operations Manager and IBM Tivoli Security Compliance Manager can be deployed effectively. It also introduces the centralized IBM Tivoli Custom Auditing and Reporting Services.

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

The team that wrote this redbook

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

The team that wrote the fifth edition of this book is from left to right: Axel, Sujit, Norman, Guilherme, Daniel, Veronica, and Chris

Axel Buecker is a Certified Consulting Software I/T Specialist at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide in the 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 I/T Specialist in Software Security Architecture.

Ana Veronica Carreno is a Software Engineer in the IBM Software Group in Colombia. She has worked at IBM for several years, specializing in IT security solutions, helping her customers to approach their infrastructure requirements and map them to business solutions. Ana Verónica holds a degree in Electronic Engineering from Universidad de los Andes, Bogota Colombia.

Norman Field is an Advisory Software Engineer at IBM Software Group. He has served as the lead interaction designer for Tivoli Federated Identity Manager since the product's inception four years ago and is also the test lead for the Tivoli Access Manager for Enterprise Single Sign-on. He has extensive experience working with customers to translate business requirements into usable technology. He has written white papers and field guides detailing how to solve customer problems with IBM software. A resident of Santa Cruz, CA, Norman holds a Masters of Science Degree in Applied Math from the University of California, Los Angeles.

Christopher Hockings is a Senior IT Specialist within the IBM Software Group. He was originally a software developer with the DASCOM company (original creators of Tivoli Access Manager) and brought into IBM as part of that acquisition. He is a team leader of a group that provides Tivoli Security pre-sales, post-sales lab services, and Level 3 support for Tivoli Access Manager. He has extensive experience working with customers in designing, deploying, and supporting Tivoli Security products. He has published papers within the IBM developer domain and is the moderator of the Tivoli Security discussion forum. A resident of the Gold Coast, Australia, Christopher holds a Bachelor of Information Technology and a Bachelor of Electronic Engineering from the Queensland University of Technology, Brisbane, Australia.

Daniel Kawer is an IBM Software Solutions Architect working for GBM Corporation, an IBM Alliance Company, located in San José, Costa Rica. He holds a degree in computer science from the Costa Rica Institute of Technology. He has a few years of experience in a variety of fields and his area of expertise is design and architecture of enterprise security solutions.

Sujit Mohanty is a Senior Security Specialist with IBM Tivoli specializing in Tivoli Security Operations Manager and ISS products. He has spent a number of years in the area of information security research in a variety of areas (anti-virus, IDPS, vulnerability scanning, and SIEM technologies). He attended the University of Virginia for undergraduate study in systems engineering.

Guilherme Monteiro is an IT Security Architect with Companhia de Sistemas, an IBM Business Partner in Brazil, with a strong focus on security solutions. He has been involved with IBM security solutions since 1999, implementing directories, access management, identity management, directory integration, risk management, and developing custom solutions for key Brazilian private companies. His company also has a strong position in Linux® security solutions, with several successful implementations on this platform.

We extend our thanks to the teams who contributed to this IBM Redbooks publication. The first edition team included:

Oleg Bascurov
Cynthia Davis
Stefan Fassbender
Mari Heiser
Rick McCarty
Guy Moins
Julien Montuelle
Jim Whitmore

The second edition team included:

Andrew Gontarczyk
Mari Heiser
Santosh Karekar
Patricia Saunders
Matteo Taglioni

The third/fourth edition team included:

Juliana Medeiros Destro
Michael Ferrell
Guilherme Monteiro
Erik Wilson

The following people helped to launch this project:

Phil Billin
David K. Jackson
Daniel Kipfer
Klaus Oberhammer
Larry Shick
Jim Whitmore

The following people contributed to the fifth edition of this redbook:

Peter Szczepankiewicz, Keith Sams, Sridhar Muppidi, Max Rodriguez, Mike Gare, Jason Todoroff, Daniel Pitre, Ori Pomerantz
IBM U.S.

Eddie Hartman
Johan Varno
IBM Norway

Become a published author

Join us for a two-to-six week residency program! Help write an IBM Redbooks publication 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 will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

ibm.com/redbooks/residencies.html

Comments welcome

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6014-04
for *Enterprise Security Architecture Using IBM Tivoli Security Solutions*
as created or updated on August 6, 2007.

August 2007, Fifth Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

We added information for the following IBM Tivoli products and components:

- Tivoli Access Manager for Enterprise Single Sign-On Version 6.0 is covered in the following chapters:
  - Chapter 15, “Access Manager for Enterprise Single Sign-On” on page 449.
  - Chapter 16, “Tivoli Access Manager for Enterprise Single Sign-On scenario” on page 491.
- Tivoli Identity Manager Express Version 4.6 is covered in the following chapters:
  - Chapter 20, “Identity Manager Express structure and components” on page 613.
- Tivoli Federated Identity Manager Business Gateway Version 6.1 is introduced in the following chapter:
  - The information about the new Federated Identity Manager Business Gateway is incorporated throughout the chapters in Part 4, “Managing federations” on page 653.

- Tivoli Security Operations Manager Version 3.1 is covered in the following chapters:
  - Chapter 28, “Security Operations Manager topology and infrastructure” on page 857.
  - Chapter 29, “Building a security information event management system” on page 889.
- An appendix has been added with productivity and functional enhancements for Tivoli Identity Manager and Tivoli Access Manager.

Changed information

We updated information for the following IBM Tivoli products:

- Tivoli Security Compliance Manager Version 5.1.1
- Tivoli Directory Integrator Version 6.1.1
- Tivoli Identity Manager Version 4.6
- Tivoli Privacy Manager has been removed

We reorganized some of the chapters that discuss the Tivoli Access Manager components in conjunction with the federated single sign-on principles.

We extended the discussion on single sign-on technologies and moved it from the appendix to Part 1, “Terminology and infrastructure” on page 1.

September 2006, Fourth Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

Changed information

This fourth edition focuses solely on updating information pertaining to Tivoli Access Manager for Operating Systems Version 6.0 in the following chapters:

- Chapter 12, “Access Manager for Operating Systems” on page 381.
- Chapter 13, “Access Manager for Operating Systems business scenario” on page 413.

April 2006, Third Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

We added information for the following IBM Tivoli products and components:

- Tivoli Security Compliance Manager Version 5.1 is covered in the following chapters:
  - Chapter 30, “Compliance management with Tivoli Security Compliance Manager” on page 903
  - Chapter 31, “Tivoli Security Compliance Manager scenarios” on page 925
- Tivoli Federated Identity Manager Version 6.0 is introduced in the new Part 4, “Managing federations” on page 653, which includes the following new chapters:
  - Chapter 22, “Business context for identity federation” on page 655
  - Chapter 23, “Federation concepts” on page 679
  - Chapter 24, “Federated Identity Manager” on page 721
  - Chapter 26, “Tivoli Federated Identity Manager patterns” on page 803
- The IBM Tivoli Common Auditing and Reporting Service, which is a component that is shipping with Tivoli Access Manager for e-business, is introduced in Chapter 27, “Introducing IBM Tivoli Common Auditing and Reporting Service” on page 845.

Changed information

We updated information about the new versions for the following IBM Tivoli products:

- Tivoli Directory Server Version 6.0
- Tivoli Directory Integrator Version 6.0
- Tivoli Access Manager for e-business Version 6.0
- Tivoli Identity Manager Version 4.6

We also reorganized the chapters that discuss the Tivoli Access Manager components.

In addition to the product specific changes, we moved the discussion about MASS into Appendix A, “Method for Architecting Secure Solutions” on page 947, and added a discussion about single sign-on in Appendix 4, “Single sign-on technologies” on page 149.

Part 1

Terminology and infrastructure

In Part 1 we introduce the business context for enterprise IT security drivers and take a closer look at architectures and network models. We also use the infrastructure context in this part to introduce directory technologies and discuss the IBM Tivoli Directory Server and Directory Integrator products.

1

Business context

An *Enterprise Security Architecture* is the design of the processes and technology to achieve security. A proven methodology is important for this design. All security architectures start with defining the business context, that being the balance of business drivers and acceptable risk. This business context is the result of decisions made from the analysis of internal and external factors. Security policies are the guidelines for this business context. The resulting architecture is a functional combination of process and technology to achieve the business goal within boundaries of the business context. The architecture must fit this business context for the enterprise to achieve security, and to provide legal and regulatory compliance.

1.1 Security, risk, and compliance

Security is the confidence that systems are operating as expected.

Systems can be accessed and used only within the confines of the business rules. Vulnerabilities are protected from exploit. This is commonly viewed as the security CIA triad, as in confidentiality, integrity, and availability. Data is disclosed only to those authorized to use it (confidentiality). Data is not modified in an unexpected or unauthorized manner (integrity). Data is available when needed by the system (availability). Some may add non-repudiation to this triad as a fourth pillar of security. Non-repudiation is the ability to determine the entity that performs an activity on the data such as the actual sender of a message. It also extends to include that data is protected in transit. Non-repudiation is usually implemented with some combination of encryption and digital signatures. Though we treated these as data centric, they are also easily treated as system or functional qualities.

Security is viewed as the boundary of acceptable risk for the organization.

The security implementation is based on the analysis of risks, and how to mitigate them. It is important to include current business drivers and the risks they pose in this analysis. The risk analysis determines the balance point between risk and benefit to the business. This analysis provides the business context.

Compliance proves that systems operate according to security expectations.

This includes operation within the boundaries of acceptable risk and within the business context. The business context includes laws and regulations, which can result in a potentially different definition of security and compliance for every enterprise. The differences result from both the methods and the factors used in analyzing the business context. The commonality is that the security policy can be defined by determining acceptable risk, how to achieve (control, mitigate or accept) that risk, and how to verify that the security is implemented as specified (compliance).

The business context defines the security policies, which are usually organized hierarchically, starting with a top level organizational security policy. This organizational policy provides broad guidance on the organizations priorities and concerns for security. The next level consists of more fine grained policies to implement the top level policy, and it may consist of several layers of policies itself. At this point the policies start to define technology requirements at a high level. Below this level you can find procedures and practices describing the technical and process details to implement the security policies. You can find more details in 1.5, “Security policies” on page 11.

An organization's security maturity impacts the amount of risk analysis necessary to define the business context and design security architecture. This maturity involves security policies and how applicable they are to the design. Regardless of that maturity some degree of analysis is needed to determine the risk and resulting policies, procedures, processes, and technology for the security architecture.

Security requirements can be categorized with guidance from established sources into a set of functions, or providing specific funtionality. These can be used to define the components and services necessary for a security architecture.

Many techniques exist for identifying and analyzing risks and determining mitigation. Guidance is needed as to the areas to consider when working towards a security policy. This guidance is often found in the British Standard 7799 (BS7799). Although there might be other ways of addressing enterprise security, we take a closer look at BS7799 to present the enormous scope of this task.

1.2 The BS7799 security standard

The British Standard 7799[1] is the most widely recognized security standard in the world. The last major publication was in May 1999, an edition that included many enhancements and improvements on previous versions. When republished in December 2000, it evolved into the International Organization for Standardization 17799 (ISO/IEC 17799). 17799 was republished again in 2005 as ISO/IES 17799:2005(E) with some revisions in areas covered.

BS7799 (ISO17799) is comprehensive in its coverage of security issues. It contains a significant number of control requirements, some extremely complex. Compliance with BS7799 is, consequently, a far from trivial task, even for the most security conscious of organizations. Full certification can be even more daunting.

It is therefore recommended that BS7799 is approached in a step-by-step manner. The best starting point is usually an assessment of the current position or situation, followed by an identification of the changes needed for BS7799 compliance. From here, planning and implementing must be rigidly undertaken.

This section is intended to help you understand the 10 different categories that have to be considered when applying an overall enterprise security approach.

[1] RiskServer, Security Risk Analysis, ISO17799, Information Security Policies, Audit and Business Continuity, http://www.riskserver.co.uk/

After the categories have been described briefly, we talk about the next step in the implementation of a security policy. The categories are:

1. Business continuity planning

   The objective of this section is to counteract interruptions to business activities and critical business processes from the effects of major failures or disasters.

2. System access control

   The objectives of this section are:

   a. To control access to information.

   b. To prevent unauthorized access to information systems.

   c. To ensure the protection of network services.

   d. To prevent unauthorized computer access.

   e. To detect unauthorized activities.

   f. To ensure information security when using mobile computing and tele-networking facilities.

3. System development and maintenance

   The objectives of this section are:

   a. To ensure that security is built into operational systems.

   b. To prevent loss, modification, or misuse of user data in application systems.

   c. To protect the confidentiality, authenticity, and integrity of information.

   d. To ensure that IT projects and support activities are conducted in a secure manner.

   e. To maintain the security of application system software and data.

4. Physical and environmental security

   The objectives of this section are:

   a. To prevent unauthorized access, damage, and interference to business premises and information.

   b. To prevent loss, damage, or compromise of assets and interruption to business activities.

   c. To prevent compromise or theft of information and information processing facilities.

5. Compliance

   The objectives of this section are:

   a. To avoid breaches of any criminal or civil law; statutory, regulatory, or contractual obligations; and security requirements.

   b. To ensure compliance of systems with organizational security policies and standards.

   c. To maximize the effectiveness of and to minimize interference to and from the system audit process.

6. Personnel security

   The objectives of this section are:

   a. To reduce risks of human error, theft, fraud, or misuse of facilities.

   b. To ensure that users are aware of information security threats and concerns and are equipped to support the corporate security policy in the course of their normal work.

   c. To minimize the damage from security incidents and malfunctions and learn from such incidents.

7. Security organization

   The objectives of this section are:

   a. To manage information security within the company.

   b. To maintain the security of organizational information processing facilities and information assets accessed by third parties.

   c. To maintain the security of information when the responsibility for information processing has been outsourced to another organization.

8. Computer and network management

   The objectives of this section are:

   a. To ensure the correct and secure operation of information-processing facilities.

   b. To minimize the risk of systems failures.

   c. To protect the integrity of software and information.

   d. To maintain the integrity and availability of information processing and communication.

   e. To ensure the safeguarding of information in networks and the protection of the supporting infrastructure.

   f. To prevent damage to assets and interruptions to business activities.

g. To prevent loss, modification, or misuse of information exchanged between organizations.

9. Asset classification and control

   The objectives of this section are to maintain appropriate protection of corporate assets and to ensure that information assets receive an appropriate level of protection.

10. Security Policy

    The objectives of this section are to provide management direction and support for information security.

Along with general areas to consider similar to those outlined in BS7799, there are also business drivers to consider in defining the business context.

1.3 Common business drivers

The business context is determined by identifying risks and determining the appropriate way to mitigate these risks. Those risks come from internal and external (or blended) business drivers. A list of some common business drivers that impact security is found below:

- Asset value:

  Asset value relates to the underlying value of the transaction in the system. For an e-retailer these are tangible assets. To a financial services company the asset may be the client information or other data used in transactions of the system. These are the assets behind the system processes.

- Legal or regulatory

  Legal and regulatory refers to the externally imposed conditions on the transactions in the business system, and the company. This includes the rules and policies imposed by regulatory and government agencies. The amount of regulation and steps to ensure compliance are factored in this driver. This would include privacy issues, the ability to prove the transaction initiator, and proving compliance.

- Time to market

  Time to market is an external business driver, reflecting the pressure to gain a competitive advantage by rapid implementation of the system. A short time to market may result in cutting corners, adding or delaying some security controls to meet the deadline.

- Simple to use

  Simple to use reflects the need or desire for the system to be intuitive to the user community. This is often a driver for single sign-on or federated identity to reduce the number of credentials required by users.

- Risk tolerance

  This is a measure of the organizations tolerance for risk. A firm with a low tolerance for risk commits to greater security around its business processes and systems. A risk averse firm shows well defined security and data policies.

- Complex organizational environment

  The complexity of an organization impacts how business decisions and processes are structured. A complex organization results in additional paths of communication and decision making relating to the business process and system. The complexity impacts security decisions as well.

- Mission critical (availability)

  Mission critical reflects the level of availability required for the business system. A mission critical system requires high availability to prevent a loss of revenue to the firm. A funds transfer system to allow float for the company's funds may be invisible to the average user, but highly critical to the company. This driver leads to a system that is easy to maintain and update, and also highly stable.

- Protect the corporate image

  This driver captures the firm's desire to protect its image, or brand(s). This measures the desire to protect the intangible image of the firm. The less likely the firm wants negative publicity from a security breech, the stronger this driver.

- Complex IT environment

  A complex IT environment reflects the environment on which the business system will be placed. A standalone facility just for our system represents the lowest complexity. A hosting facility with other systems, and other firms represents a more complex environment. An environment with a larger number or systems, or varied network access paths, or with complex architecture is a complex IT environment.

- Complex system

  This measure the complexity of the system itself.A complex system involves many linked applications (or systems), many different protocols, wide variation in user types, and access to many different classes of data.

- High risk IT environment

  This measure the environment's susceptibility to attacks. An environment (or system) likely to be vulnerable to attacks would be high risk. The vulnerability could be due to the underlying systems, track record of hackers attacking this type of system, or insecure networks or designs.

1.4 Risk analysis and mitigation

Every organization faces risk. That risk requires involvement of top level management to decide the major security risks for that type of business and how to address them. Risk analysis involves assessing what could go wrong, how likely it is to occur, and what damage results from that event. Elements to analyze include:

- Threats: The events, forces or persons that pose the risk. This could be an event to exploit a vulnerability.
- Probability: The likelihood this threat would occur.
- Damage: The impact of the threat being exploited. This includes loss of service, revenue, potential revenue, and image among and other business specific elements.
- Trade-offs: Evaluating two competing business drivers and evaluating the advantages and disadvantages of each to reach a compromise solution. A common technique to analyze these trade offs is a *business impact analysis*.

The areas from BS7799 and the business drivers present general guidance on areas to consider when analyzing risk. The result of the analysis is a collection of risks to the organization.

Risk mitigation is determining how to handle those risks. For each risk area, the options are to:

- Reduce: Lower the risk through controls, or technology.
- Transfer: Offload the risk by placing it on some other entity.
- Accept: Decide the risk is acceptable based on the benefit.
- Ignore: Choose not to reduce, transfer or accept the risk. This is equivalent to accepting the risk, but without due diligence.

The goal is to reduce, or eliminate, the risks identified. A security policy is a mechanism to manage risks. This policy involves a combination of process and technology to bring the risk to an acceptable level as depicted in Figure 1-1 on page 11.

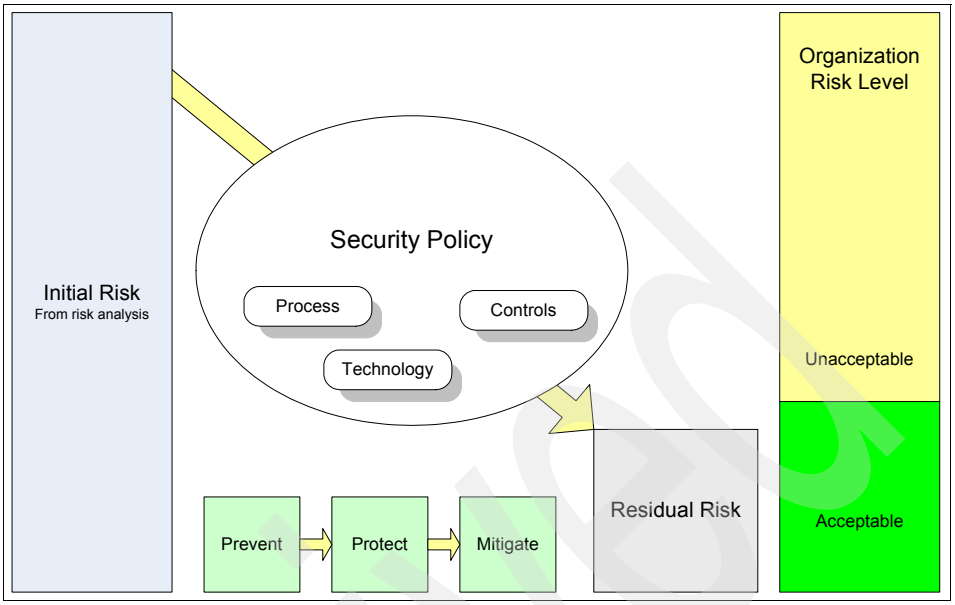


Figure 1-1 Reducing risk

In Figure 1-1 we start with the initial risk level. We see that the security policy provides guidance on controls, processes, and technology to lower the risk. The security policy prevents, protects from, or mitigates the risk reducing severity. Risk is not eliminated but reduced. An organization establishes what is the acceptable risk level, either quantitatively or qualitatively. This can be through business impact analysis, or other techniques to balance risks and benefits. For example, a business may choose to accept a higher risk to accommodate such drivers as time to market and ease of use. The security policy must reduce the residual risk to, or below the level acceptable to the organization for the security policy to be effective. This residual risk can result from an inability to further reduce the risk, or a conscious decision not to invest more resources to do so.

1.5 Security policies

A security policy is driven by the corporate decisions regarding risk based on the business context. It is the result of determining what is at risk, and how to reduce that risk. The same set of threats and risks may be viewed as less severe by a more risk accepting organization. This means that the security policy must be individually crafted.

A security policy is somewhat a misnomer as the policy is really a set of layers of policies, on top of procedures and practices. These provide the framework for the

technical side of the security architecture. It is important that a standard and proven methodology be used for risk identification, mitigation, and developing security policies. This can be handled by external consultants such as IBM Global Services.

The top layer of the security policies is the corporate security policy. It sets the high-level direction for the organization. It's scope is organization wide and represents a general statement of the security goals. This corporate policy is both static, and non-technical, being goal driven and not specifying technologies. It provides broad guidance for the organization, leaving more dynamic and technical details to lower policy layers. As shown in Figure 1-2 the next policy layer is usually inscribed to standards.

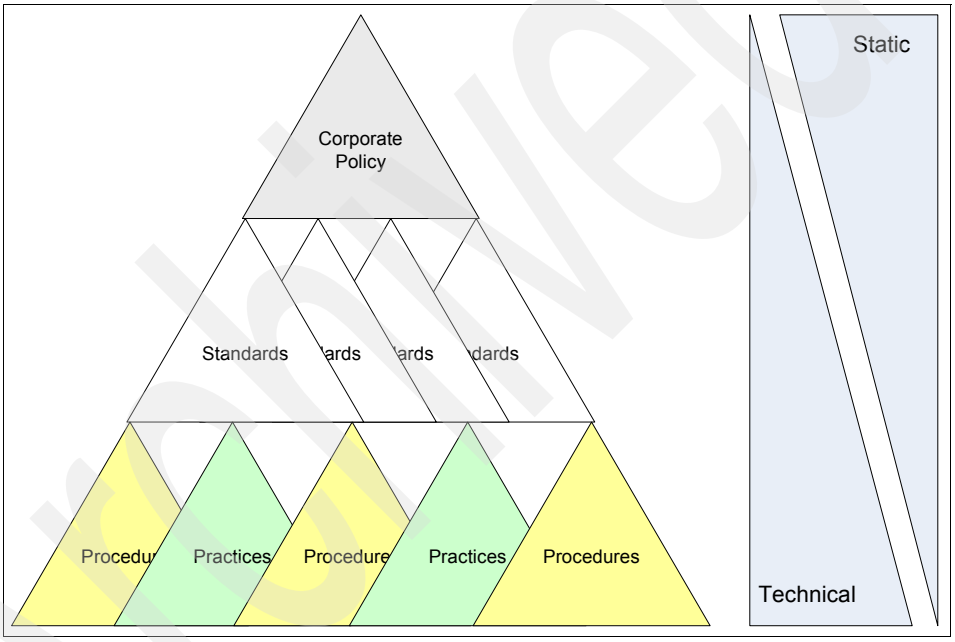


Figure 1-2 Dynamics for policy, standards, practices, and procedures

Standards take the general goals and restates them in terms of specific technology areas. Below this are practices and procedures, the most technical and dynamic layers of policies. These represent the details needed to implement the overall security policy. Practices are detailed steps to implement the technology. Procedures are steps used to interface the technology with the environment (users, operators, and so on). At this layer the procedures may specify products and specific processes to be used.

For example, one requirement of corporate policy states that authentication must occur only once for ease of use. A standard would state a more specific

requirement stating that a single sign-on technology is needed across all applications and systems. The practices and procedures would specify identity management and access control products, as well as processes to populate and manage users.

Attention: *Policy* is a very common term and in many products you will find specific *policies* sections. These are the product-related policies that are covered in the practice or procedure documents. The *corporate policy* is not related to products and is a high-level document.

1.5.1 Security policy lifecycle

The lifecycle of a security policy includes five basic steps as shown in Figure 1-3 on page 14:

- Assess risk resulting from the business context for the organization. This assessment provides the business context necessary to develop the security policies.
- Develop security policies: This is the development of the layers of policies (standards, practices and procedures) to put the security in place. These policies are communicated to the organization as needed.
- Implement security policies.: Security policies are put into effect, and used to manage normal operation.
- Manage security policies: Security policies are reviewed for effectiveness, and currency.
- Audit security policies: Audit is used to measure both the degree to which the policy is adhered to and identify any gaps in the policy. This is a logical jumping off point to re-assess the business context, looping back into the risk assessment step.

This is a common approach adopted in many methodologies. Audit results often point back to the first step to reassess business context and risk and then refine or revise the security policies. A proactive approach is to schedule regular review cycles, and adjust policies accordingly. This lifecycle applies to all levels of security policies, from the corporate level down to the practice and procedure level.

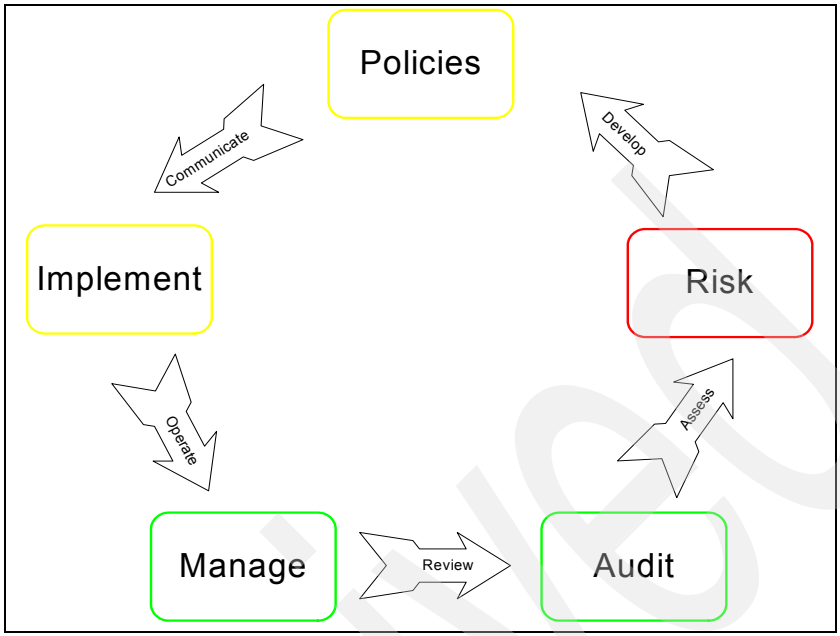


Figure 1-3 The five steps in defining your IT security

1.6 Other considerations

Let us refocus one more time and state that a security policy is written to:

- Provide enterprise-wide rules.
- Highlight risks and the way to cope with it.
- Formalize the security measures that must be applied.
- Set up the expectations between the employee and the enterprise.
- Clarify the procedures to follow.
- Provide legal support in case of problems.

A security policy provides guidance to employees and the organization as to acceptable actions and expectations relating to an organizations's Information Technology. To be successful these policies must be:

- Clear and not subject to interpretation. This is especially true where violation of policies may result in legal action.
- Published and available to those impacted.
- Reviewed and updated regularly to reflect changes in business context, risk, and regulation.

It is more efficient to get the staff enforcing a policy or standard they understand than having them fight against it. They are a key part of the global security level of the enterprise, and when they try to bypass some policy, they put your enterprise in danger.

1.6.1 The human factor impact

The most common source of security problems is employees making mistakes. The actual threat from hackers and viruses is much smaller than most people would anticipate. Figure 1-4 details the various sources.

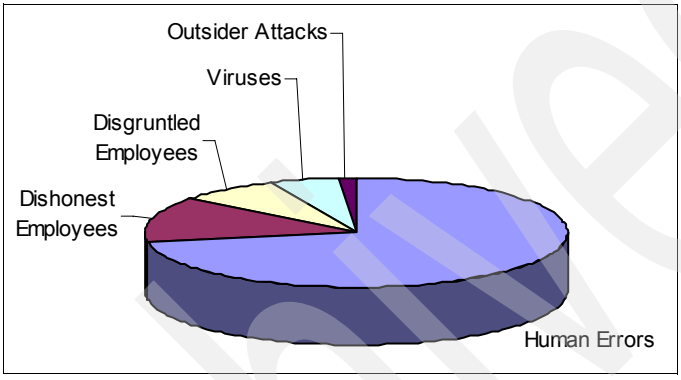


Figure 1-4 Principal threat sources

The biggest threat comes from inside. A total of 71% of problems are directly related to employees, with 55% not intending to cause damage.

Having policies and procedures in place helps you address your risks. However, they will not directly cover the human factor errors. This is where technology serves a useful role in providing security.

Managing and auditing your security enables you to perform checks and discover some errors and correct them. However, if discovered, they could have already been the cause of a security breach.

Another important factor in managing and implementing your procedures is in using computer assistance with automatic verifications in order to reduce the possibility of human errors. A good example is the management of user accounts and access rights. Even today, communication about a new employee or one transferring from one department to another is still being implemented using mail or paper. These steps, with a lot of human interaction, are the most error-prone processes, easily leading to assigning too many or wrong access rights, or even keeping an account alive for somebody who has left the enterprise.

This risk in this example, introduced by the human factor, can be partially mitigated by using a workflow, a user management tool, or both. It will be configured to apply the standards at all times. Some of these tools use workflow systems that can even implement the procedures. This will not prevent all errors but will cover a lot of them. Using a central repository also increases the global security by avoiding discrepancies between the various access control systems. The way your corporate policy and standards are applied has a direct impact on the quality and the level of security.

1.6.2 Legal and regulatory concerns

Legal and regulatory concerns must be considered when determining the business context, analyzing risk, and developing security policies. There are several well known recent regulatory guidelines impacting differing industry sectors:

- Gramm-Leach-Bliley Act (GLBA; also known as the Financial Services Modernization act). Information about GLBA may be found at either of the following Web sites:

  http://banking.senate.gov/conf/
  http://www.ftc.gov/privacy/privacyinitiatives/glbact.html

- Sarbanes-Oxley Act (SOX)

  http://sarbanes-oxley.com

- Health Insurance Portability and Accountability Act (HIPAA)

  http://www.cms.hhs.gov/hipaa

- U.K. Data Protection Act 1998

  http://www.opsi.gov.uk/acts/acts1998/19980029.htm

- European Data Directive 95/46/EC

  http://www.cdt.org/privacy/eudirective/EU_Directive_.html

- Basel II

  http://www.bis.org/publ/bcbsca.htm

Note: Customers are responsible for ensuring their own compliance with various laws and regulations such as those mentioned above. It is the customers sole responsibility to obtain the advice of competent legal counsel regarding the identification and interpretation of any relevant laws that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal, accounting or auditing advice, or represent that its products or services ensure that the customer is in compliance with any law.

1.7 Closing remarks

Enterprise Security Architecture is the design of processes and technology to achieve security. It is not merely technology, nor merely process, but a mixture. The business context sets the stage to determine risks, and developing a security policy. The implementation of this policy determines the processes, technology, and products that are needed.

We have not discussed specific steps to design the security architecture based on the policy. We approach this in Chapter 2, "Common security architecture and network models" on page 19. You need someone capable of applying a set of rules and guides to the unique facts of your enterprise: an *architect* who follows a methodology that is designed to help describe and develop a complex security architecture. IBM has developed and uses a *Method for Architecting Secure Solutions* (MASS) that reflects the current impact of thriving e-business environments. We explore this methodology as part of Chapter 2, "Common security architecture and network models" on page 19, and in more detail in Appendix A, "Method for Architecting Secure Solutions" on page 947.

Network topographies play an immensely important role for the enterprise security IT architecture, and without detailed knowledge of where to establish perimeter security components, one cannot succeed. Section 2.2, "Common network components" on page 26, talks about these aspects by laying a foundation of understanding about why the network becomes more and more critical to the overall IT infrastructure and security.

Finally, infrastructure elements are needed to provide cross-system services. A directory is one of these components that cannot be mapped into one distinct category but offers a broad spectrum of capabilities. Chapter 3, "Directory technologies" on page 49, addresses these capabilities.

2

Common security architecture and network models

So far we have established how to develop security policies based on risk analysis of the business context. This chapter moves us to designing the security architecture. We have indicated that solutions vary based upon the business context and decisions about risk for each organization. There are though common security subsystems that can be leveraged in developing the security architecture. There are also common network models that are used to provide security in the infrastructure.

2.1 Common security architecture subsystems

Chapter 1, “Business context” on page 3, showed us that the business context, risk analysis and mitigation, and security policies may be unique for every organization. This does not preclude models for security. A similar model may be used for several organizations, with differences found in configuration and procedures around the technology to meet the unique goals of the organization’s security policy. This makes the task of designing a security architecture somewhat less daunting.

There also exists a set of common subsystems for providing functional security services. These are identified in the IBM developed *Method for Architecting Secure Solutions* (MASS). The MASS methodology is grounded on Common Criteria, which is located in Appendix A, “Method for Architecting Secure Solutions” on page 947. It represents one method for security architecture, and its subsystems are used throughout this book.

2.1.1 Common Criteria

The MASS methodology was designed after careful evaluation of security standards, such as BS7799, and the Common Criteria. These standards represent internationally accepted “best practices” for design and measurement of security, but do not specify specific technologies or products.

Common Criteria provide a taxonomy for evaluating security functionality through a set of functional and assurance requirements. The Common Criteria include 11 functional classes of requirements:

- Security audit
- Communication
- Cryptographic support
- User data protection
- Identification and authentication
- Management of security functions
- Privacy
- Protection of security functions
- Resource utilization
- Component access
- Trusted path or channel

These 11 functional classes are further divided into 66 families, each containing a number of component criteria. There are approximately 130 component criteria currently documented, with the recognition that designers may add additional component criteria to a specific design. There is a formal process for adopting

component criteria through the Common Criteria administrative body, which can be found at:

http://csrc.nist.gov/cc/

While these classes and their families represent functional areas for requirements, they do not afford the economy and reusability of specific security subsystems offering services.

2.1.2 MASS security subsystems

A security architecture must be designed through a consistent and proven methodology. MASS was designed to abstract the functional classes of the Common Criteria into an aggregation that reflects a small group of security functions, which may (and usually do) interact with other function groups.

An analysis of the 130 component-level requirements in relation to their function within an NIS solution suggests a partitioning into five operational categories:

- Audit
- Access control
- Flow control
- Identity and credentials
- Solution integrity

A summary mapping of CC classes to functional categories is provided in Table 2-1. Realize that this is not a one-to-one mapping—a single CC functional class (for example, *data protection*) may appear in more than one MASS functional category (for example, *access control, flow control, identity/credential,* and *solution integrity*).

Table 2-1 Placing Common Criteria classes in functional categories

Functional category	Common Criteria functional class
Audit	Audit, component protection, and resource utilization
Access control	Data protection, component protection, security management, component access, cryptographic support, identification and authentication, communication, and trusted path/channel
Flow control	Communication, cryptographic support, data protection, component protection, trusted path/channel, and privacy
Identity/credentials	Cryptographic support, data protection, component protection, identification and authentication, component access, security management, and trusted path/channel

Functional category	Common Criteria functional class
Solution integrity	Cryptographic support, data protection, component protection, resource utilization, and security management

These five subsystems may be meshed working interactively with each other in a security architecture as shown in Figure 2-1. The interaction may be communication between various products in each subsystem or processes to link, or manage the interaction.

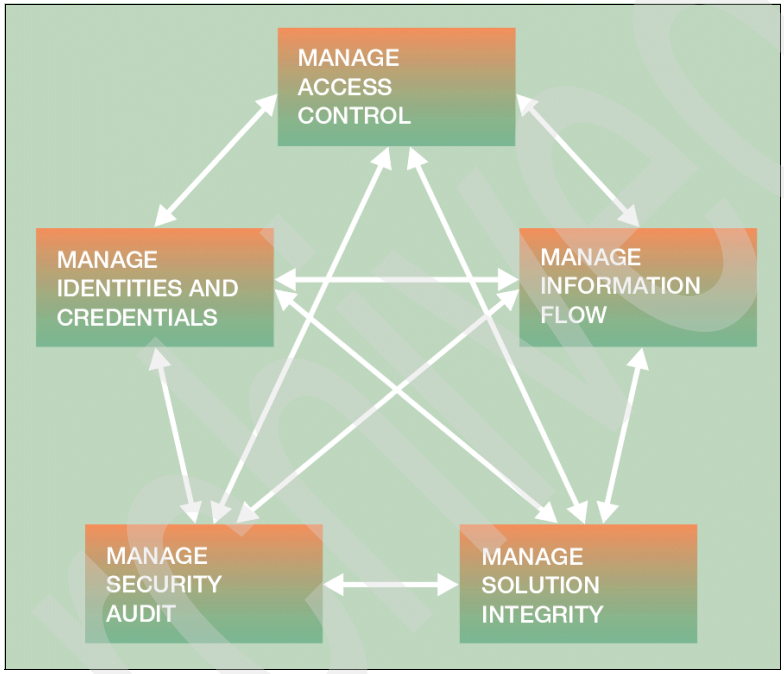


Figure 2-1 IT security processes and subsystems

A brief description of each of the five security subsystems is provided. A more detailed description, along with how these subsystems aggregate the Common Criteria functions and classes, is in Appendix A, “Method for Architecting Secure Solutions” on page 947. Each of these subsystems may interact with the other subsystems to provide the security solution.

Security audit subsystem

The purpose of this subsystem is to provide proof of compliance to the security policy. A security audit subsystem is responsible for capturing, analyzing, reporting, archiving, and retrieving records of events and conditions within a computing solution. Security audit analysis and reporting can include real-time

review, as implemented in intrusion detection components, or after-the-fact review, as associated with forensic analysis in defense of repudiation claims. The security audit subsystem provides:

- Collection of security audit data, including capture of the appropriate data, trusted transfer of audit data, and synchronization of chronologies
- Protection of security audit data, including use of time stamps, signing events, and storage integrity to prevent loss of data
- Analysis of security audit data, including review, anomaly detection, violation analysis, and attack analysis using simple heuristics or complex heuristics
- Alarms for loss thresholds, warning conditions, and critical events

Solution integrity subsystem

The purpose of the solution integrity subsystem in an IT solution is to address the requirement for reliable and correct operation of a computing solution in support of meeting the legal and technical standard for its processes. The solution integrity subsystem may rely on the audit subsystem to provide real-time review and alert of attacks, outages, or degraded operations, or after-the-fact reporting in support of capacity and performance analysis. The solution integrity subsystem provides:

- Integrity and reliability of resources
- Physical protection for data objects, such as cryptographic keys, and physical components, such as cabling, hardware, and so on
- Continued operations including fault tolerance, failure recovery, and self-testing
- Storage mechanisms: cryptography and hardware security modules
- Accurate time source for time measurement and time stamps
- Prioritization of service via resource allocation or quotas
- Functional isolation using domain separation or a reference monitor
- Alarms and actions when physical or passive attack is detected

Access control subsystem

The purpose of an access control subsystem in an IT solution is to enforce security policies by gating access to, and execution of, processes and services within a computing solution via identification, authentication, and authorization processes, along with security mechanisms that use credentials and attributes. The credentials and attributes used by the access control subsystem along with the identification and authentication mechanisms are defined by a corresponding credential subsystem. The access control subsystem may feed event information to the audit subsystem, which may provide real-time or forensic analysis of

events. The access control subsystem may take corrective action based on alert notification from the security audit subsystem. The access control subsystem provides:

- Access control enablement
- Access control monitoring and enforcement
- Identification and authentication mechanisms, including verification of secrets, cryptography (encryption and signing), and single-use versus multiple-use authentication mechanisms
- Authorization mechanisms, to include attributes, privileges, and permissions
- Access control mechanisms, to include attribute-based access control on subjects and objects and user-subject binding
- Enforcement mechanisms, including failure handling, bypass prevention, banners, timing and timeout, event capture, and decision and logging components

Information flow control subsystem

The purpose of an information flow control subsystem in an IT solution is to enforce security policies by gating the flow of information within a computing solution, affecting the visibility of information within a computing solution, and ensuring the integrity of information flowing within a computing solution. The information flow control subsystem may depend on trusted credentials and access control mechanisms.

This subsystem may feed event information to the security audit subsystem, which may provide real-time or forensic analysis of events. The information flow control subsystem may take corrective action based on alert notification from the security audit subsystem. The information flow control subsystem provides:

- Flow permission or prevention
- Flow monitoring and enforcement
- Transfer services and environments: Open or trusted channel, open or trusted path, media conversions, manual transfer, and import to or export between domains
- Observe mechanisms: To block cryptography (encryption)
- Storage mechanisms: Cryptography and hardware security modules
- Enforcement mechanisms: Asset and attribute binding, event capture, decision and logging components, stored data monitoring, rollback, and residual information protection and destruction

Identity and credential subsystem

The purpose of a credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms, the processes, and the security subsystems within a computing solution. In some applications, credential systems may be required to adhere to legal criteria for creation and maintenance of a trusted identity used within legally binding transactions.

A credential subsystem may rely on other subsystems in order to manage the distribution, integrity, and accuracy of credentials. A credential subsystem has, potentially, a more direct link to operational business activities than the other security subsystems, owing to the fact that enrollment and user support are integral parts of the control processes it contains. A credential subsystem may include the following functional requirements:

- Single-use versus multiple-use mechanisms, either cryptographic or non-cryptographic
- Generation and verification of secrets
- Identities and credentials to be used to protect security flows or business process flows
- Identities and credentials to be used in protection of assets: integrity or non-observability
- Identities and credentials to be used in access control: identification, authentication, and access control for the purpose of user-subject binding
- Credentials to be used for purposes of identity in legally binding transactions
- Timing and duration of identification and authentication
- Lifecycle of credentials
- Anonymity and pseudonymity mechanisms

Summary of the security subsystems model

The five security subsystems described here exist within every IT solution at the conceptual level. The subsystems are aggregations of process and technology to achieve security functionality. A single component may perform a function, or interface with some or all of the subsystems. A subsystem may be distributed amongst several components in an environment. All IT solutions require security functions related to these subsystems. Security architecture is the design, integration, and networking of the services and mechanisms associated with these subsystems to provide security functionality.

2.2 Common network components

Networks are the mechanism for electronic communication between systems. Just as we have found there are common security architecture subsystems, we see common models of networks. The view of the network and security has changed over time. Network security used to be focused on hard boundaries, with limited access to and from the internet. Now networks must provide a variety of communications in and out of an organization in a carefully controlled manner. There must be a balance between blocking malicious traffic, and allowing traffic in a controlled manner.

Networks are the foundation for e-business. They must be functioning in a secure manner aligned with the business context. This means the network structure must consider risks and mitigate them through its design. The basic security architecture methods we've discussed of analyzing the business context, assessing and mitigating risks apply to network design. We will see that this evolves to some common network models for setting the level of security in different network zones.

2.2.1 Building network boundaries

Network boundaries are used to isolate networking zones with differing security policies. These boundaries are created to implement restrictions on the type of traffic that is allowed in a zone. An example might be to restrict access to only http traffic on port 80 and HTTPS traffic on port 443 inbound from the outside to a zone of Web servers. We use a firewall to allow this traffic and block all others. In its simplest case, a firewall is a device that implements a policy regarding network traffic. It creates boundaries between two or more networks and stands as a shield against unwanted penetrations into your environment. But as in construction terms, it is not meant to be your only line of defense, rather a mechanism to slow the progress of an intrusion.

One method of shielding information about the network the firewall protects is through re-addressing the packets so that outbound traffic appears to have originated from an address associated with the firewall itself. This re-addressing is called *Network Address Translation* (NAT) and its primary function is to hide the trusted network from untrusted networks.

Firewalls may be bundled with other features such as content filtering, Virtual Private Network (VPN) functionality, and even authentication. We will deal with firewalls without exploring these features. The next few sections describe several basic firewall types and how they function.

Packet filter firewall

A *packet filter firewall* uses a rule set to decide what traffic should be allowed, and what should be blocked. It does this by analyzing individual network packets, and matching them to a set of predefined rules. The packet filter will either allow or disallow communication based on the information in the packet and the direction it is heading. Elements that are evaluated against the rules are the physical network interface the packet arrives on, the IP address the data is coming from, the address the data is going to, the type of transport protocol being used (UDP, TCP, ICMP), the source port, and the destination port.

This type of firewall is very simplistic and does not look at the packet's application layer data and does not track the state of the connection. It allows access through the firewall with the least amount of inspection. Because it is simplistic, it is the fastest firewall technology available.

Circuit level firewall

Circuit level firewalls confirm that a packet is either a connection request or a data packet belonging to a connection. To validate the connection, the circuit level firewall examines each connection to ensure that it offers a legitimate handshake for the protocol being used. Data packets are not forwarded until the process is complete.

This type of firewall stores information as dynamic rules regarding that connection. These are in the form of a virtual state table about the session at the transport layer. All incoming packets are compared against rules on the transport layer. If the packet meets all conditions of the circuit table and rules, it is allowed.

Application layer firewall

Application layer firewalls examine the information in network packets but operate at the application level. They view information as a data stream and not as a series of packets; therefore, they are able to scan information being passed over them and ensure that the information is acceptable based on their set of rules and logic. This allows the firewall to make some intelligent decisions about what to do with packets that pass through it.

Application layer firewalls generally take the form of specialized software and proxy services, allowing no traffic directly between networks. They also have the added feature of performing logging and auditing of traffic passing through them. This enables them to communicate with an *intrusion detection system* (IDS) and log information regarding an attack.

Dynamic packet filter firewall

Also referred to as stateful inspection, dynamic packet filtering does not examine the contents of each packet. Instead, it compares certain key parts of the packet to a database of trusted information. Information traveling from inside the firewall to the outside is monitored for distinctive characteristics, then incoming information is compared to these characteristics. If the comparison yields a reasonable match, the information is allowed through. Otherwise it is discarded.

The dynamic packet filter acts at the network layer, and tracks each connection negotiating all interfaces of the firewall to ensure that they are valid. It also monitors the state of the connection and compiles the information in a state table. Because of this, filtering decisions are based not only on administrator-defined rules (as in static packet filtering) but also on context that has been established by prior packets that have passed through the firewall. It also has an added security measure with which it closes off ports until connection to the specific port is requested. This is an effective counter to port scanning.

Routers

A router is an interconnection device that links discrete networks and forwards packets between them. A router makes decisions on whether to forward a packet between networks based on a configuration table of routes, and addresses information in a packet. A router may be used to isolate the networks from one another, preventing the traffic on one from unnecessarily spilling over to the other. Why discuss routers within the context of firewalls? The two usually work in conjunction with each other. A solid firewall installation uses a combination of the technologies offered by routing and filtering. Figure 2-2 outlines a basic firewall installation.

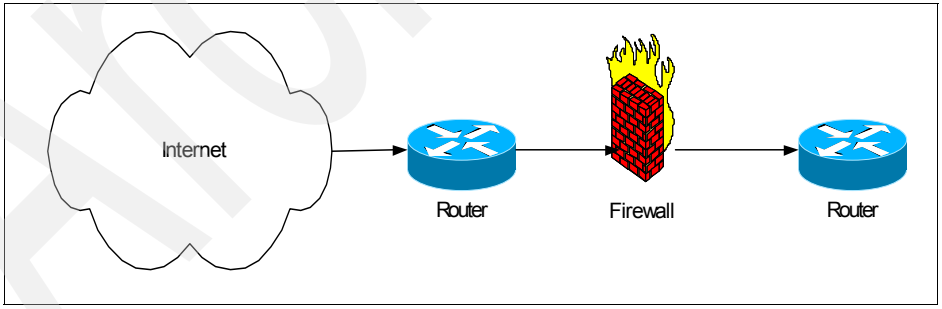


Figure 2-2 Basic Internet boundary network configuration

2.2.2 Intrusion detection and prevention

A discussion about the network would not be complete if we did not look at intrusion detection and prevention devices. We can regard those as necessary

components of the security audit and solution integrity subsystems. Intrusion detection and prevention are available for both network and hosts. Here we focus on networks, but similar technologies exist for individual hosts as well.

Network intrusion detection systems (NIDS) monitor network traffic for unwanted or improperly formatted traffic. This unwanted traffic could be between systems in a network zone, or from the Internet into the network. Network sensors monitor the traffic in a passive mode, logging packet information based upon rules in the sensor. These rules can trigger alerts when suspicious or unwanted traffic occurs.

The major drawback with intrusion detection devices comes from the sheer volume of data produced. Every packet flowing past the sensor may be logged. Filters and rules may reduce this, but regardless, NIDS produce a large amount of data. Since a network may have several of these devices, data aggregation also becomes a problem. Another issue is to find the balance between alerts that view normal traffic as suspicious (false positives) and viewing suspicious traffic as normal (false negatives). Finding the balance is called “tuning” the NIDS. NIDS have matured in recent years to provide better data aggregation, and to help reduce the effort to for tuning.

There are two distinct types of NIDS as well, signature based and heuristic. Signature based NIDS require individual rules to be constructed for types of traffic to either monitor or ignore. The rules tell the NIDS how to view traffic. Heuristic based NIDS use statistical or algorithmic techniques to determine what is normal traffic and what is suspicious. The advantage is alerts are based on traffic patterns, and this allows for a more dynamic configuration of what is normal, and what is not. Many claim this provides an advantage in day zero virus incidents as unusual traffic activity is more likely to be detected in a heuristic based NIDS.

A growing interest is placed on extending the detection of suspicious traffic to a method to prevent it. This is the role of *network intrusion protection systems*. These devices trigger an action to eliminate suspicious traffic when detected. Of concern is the impact if traffic viewed as suspicious is blocked, but the traffic is actually permissible.

As mentioned above, similar technologies exist for individual hosts on the network. These include detection through signatures or heuristics, as well as prevention.

2.3 Common network models

There are common network models for security architectures, with the similar security requirements being grouped into zones. We start by looking at the business context, and the various components. Building an architectural model that represents key components and the connections or interfaces between components allows for a visual picture of the business needs, as shown in Figure 2-3.

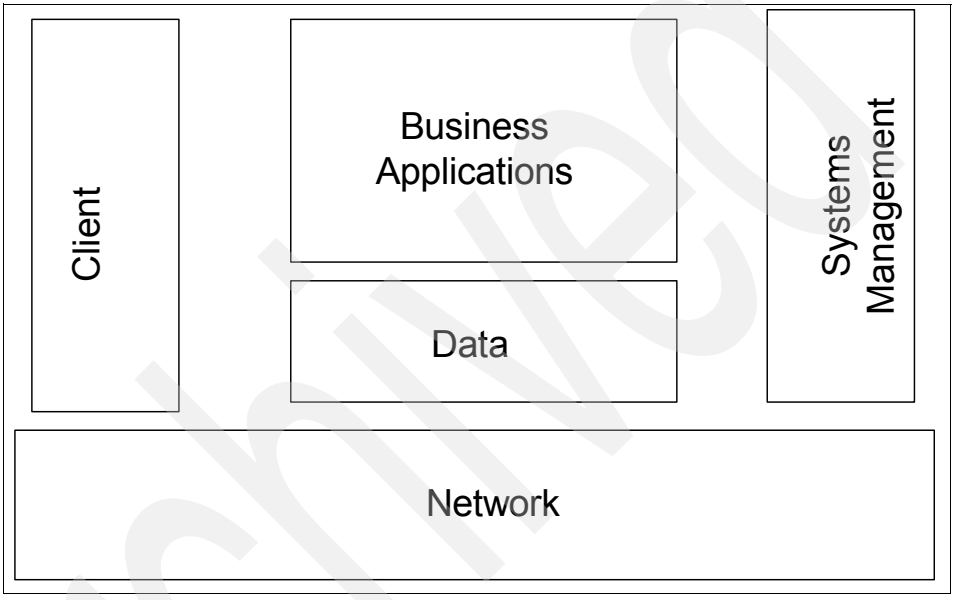


Figure 2-3 High-level architectural model including all network components

Looking at the enterprise in this manner gives you the opportunity to visualize the relationships among your basic systems. It should also enable you to drill down into each component for the visualization of additional relationships.

Perhaps the most important relationship, in terms of this discussion, is that security no longer comprises simply the network but surrounds the entire enterprise, as depicted in Figure 2-4 on page 31.

Notice that this model incorporates the client. That action opens the door to realizing that the Web is the network of organizations, where the traditional client server model is now multi-dimensional and the security concerns are immediately more complex. The user population increases geometrically, identification of users and hosts accessing data is no longer easy, and controlling access and availability becomes a major concern. The security needed to protect your environment must evolve as well.

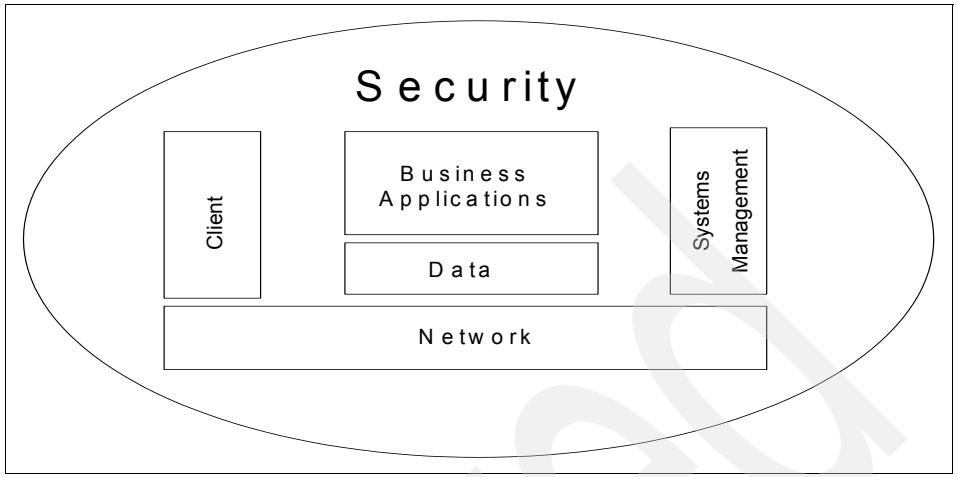


Figure 2-4 How security fits into the enterprise

Does the evolution of your security requirements mean that you abandon the methods you have used to date? Do you simplify the components by limiting your approach to firewalls and antivirus software? No, it simply means that you must globalize your security and localize your approach. Adopting, installing, and independently managing different technologies from multiple vendors in many locations will not give you the reduced time-to-market required in the e-business environment or cost-effective management of your enterprise.

2.3.1 Localizing a global vision

A global vision suggests that the enterprise is more than its physical boundaries. But localizing that perspective tames the complexity of trying to install, implement, and manage a security solution. To achieve this, you can base the solution on an integrated, standards-based architecture. An open and adaptable architecture helps reduce unseen flaws that can compromise the entire infrastructure and reduce the availability of applications and information.

Adding security design objectives into your architecture creates a framework to organize and validate the business environment and security risks. The immediate benefit is saved time and lower costs to reach the outcome. However, using a tried methodology gives a better-quality result with a quantitative tracking method. Security design objectives should outline how to achieve the following:

- Deploy and manage trusted credentials.
- Control access to stored information consistent with roles, responsibilities, and privacy policies.

- Control access and use of systems and processes consistent with roles and responsibilities.
- Protect stored or “in transit” information consistent with its classification, control, and flow policies.
- Assure the correct and reliable operation of components and services.
- Defend against attacks.
- Defend against fraud.

The IBM Method for Architecting Secure Solutions (MASS), discussed more in Appendix A, “Method for Architecting Secure Solutions” on page 947, provides you with design objectives or, more simply put, a starting point. MASS provides a set of security domains to help define the threats to an enterprise (including actors and users, flow control, authorization, physical security, and so on). It enables you to assign information assets to your security domains that become crucial in high-level designs of your architecture. We will use the MASS developed security zones throughout the book.

Using Figure 2-5, think of these areas as uncontrolled, controlled, restricted, secured, and external controlled. The client utilizes the network to access applications and data. This client can be from either within your enterprise or outside of it. Using the concept of security domains you can translate Figure 2-5 into something more targeted, as shown in Figure 2-7 on page 37.

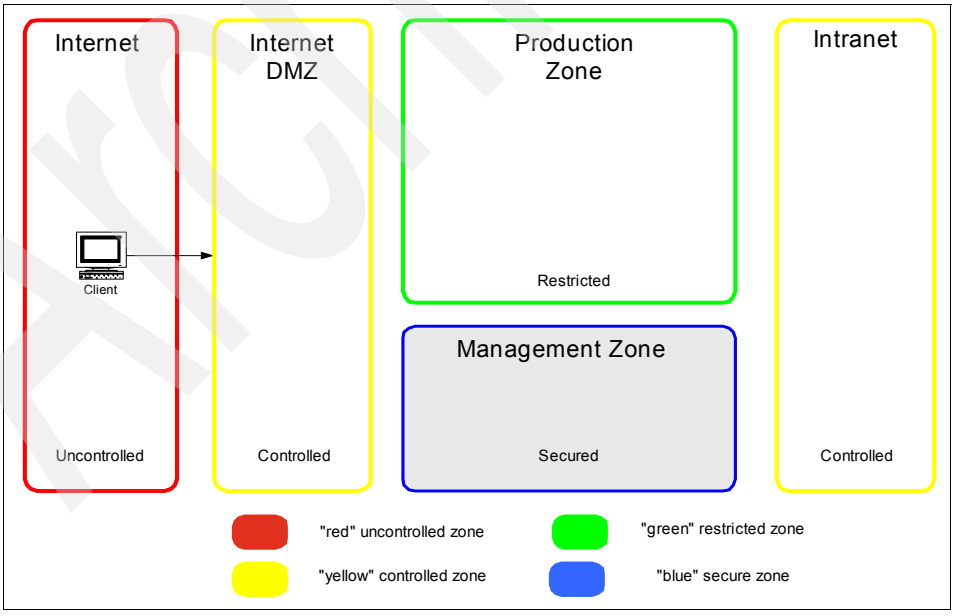


Figure 2-5 Applying MASS domain concepts

Note: The breaks between each network zone indicate the use of a firewall that clearly delineates each perimeter from the next.

Let us briefly explain what these domain categories stand for:

Uncontrolled	Refers to anything outside the control of an organization. Access from the uncontrolled environment to systems in the controlled zone could be via a wide number of channels.
Controlled	Restricts access between uncontrolled and restricted (a traditional DMZ).
Restricted	Access is restricted and controlled. Only authorized individuals gain entrance and there is no direct communication with external sources (Internet).
Secured	Access is available only to a small group of highly trusted users. Access to one secured area does not necessarily give access to another.
External controlled	An external zone in which data is stored by business partners external to the systems where there is limited trust in the protection of data (for example, credit reporting agencies, banks, and government agencies).

Constructing your environment on this manner enables internal users to “see” out, but external users cannot “see” in. The external users access is restricted. The benefits of constructing security domains this way are:

- They are clear and efficient.
- They are easy to explain.
- They are easy to work with.
- They provide a complete design and implementation view, enabling you to avoid errors.
- Fewer errors mean a lower risk of exposure and loss.
- Consistent use of recognized standards (Common Criteria, IBM Architecture Description Standard).

MASS uses the Common Criteria definition of risk management as a framework, identifying four steps in risk management:

- Identification of vulnerabilities
- Identification of threats or threat agents
- Determination of the risk imposed
- Identification of available countermeasures

Security risk management plays a big part in designing a secure solution, but so does security assurance. If we define the risks to the system we must also assure that we countermeasure those risks providing assurance for the correctness and effectiveness of the security solution.

You will see these domain designs throughout the book. Figure 2-7 on page 37 and Figure 2-8 on page 38 have clearly marked firewalls to help you become familiar and comfortable with the placement and domain concept.

2.3.2 Network zones

We have to consider four types of network zones and their transport classifications in our discussion:

- Uncontrolled (the Internet)
- Controlled (an Internet-facing DMZ and the intranet)
- Restricted (a production network)
- Secure (a management network)

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*.

Other networks

Keep in mind that the network examples we use do not necessarily include all possible situations. There are organizations that extensively segment functions into various networks. However, in general, the principles discussed here may be translated easily into appropriate architectures for such environments.

Placement of various data components within network zones is both a reflection of the security requirements in play and a choice based on an existing or planned network infrastructure and levels of trust among the computing components within the organization. Requirement issues often may be complex, especially with regard to the specific behavior of certain applications. With a bit of knowledge about the organization's network environment and its security policies, reasonable component placements are usually easy to identify.

Figure 2-6 on page 36 summarizes the general component-type relationships and the transport classifications to the network zones discussed above.

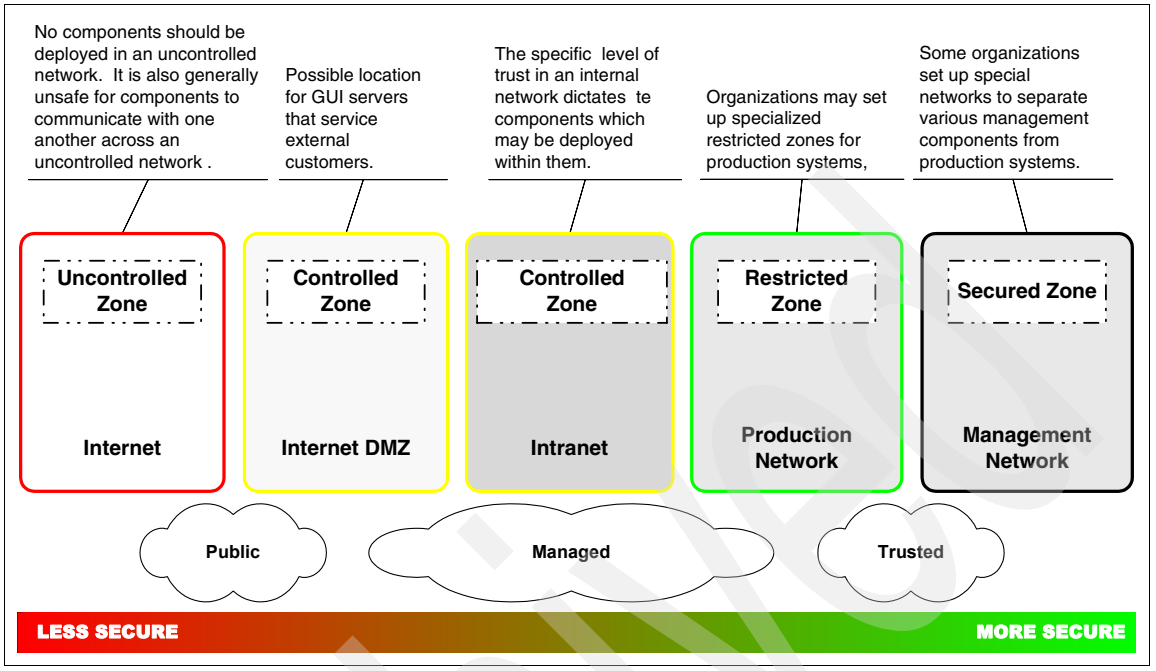


Figure 2-6 Graphic representation of network zones, transport classifications, and their level of trust

2.3.3 E-business security requirement

The IBM e-business methodology fits nicely with MASS domain concepts. E-business patterns originate in IBM product divisions and are provided as operational models that are also based on open standards and technologies. Notice that the principles of the “Six A’s” of e-business factor nicely into the overall plan as well:

Authorization Allowing only users who are approved to access systems, data, application, and networks (public and private).

Asset protection Keeping data confidential by ensuring that privacy rules are enforced.

Accountability Identifying who did what, when.

Assurance The ability to confirm and validate the enforcement of security.

Availability Keep systems, data, networks, and applications reachable.

Administration Define, maintain, monitor, and modify policy information consistently.

In order for your network security solution to work, it must be based on consistent, corporate-wide policies. A successful deployment requires that an effective link be forged from the management definition of policy to the operational implementation of that policy.

Tip: Plan your security polices around your business model, not the other way around. For more information about corporate policies, see 1.5, “Security policies” on page 11.

2.4 Practical designs

The DMZ, or outermost perimeter network, is the separation point between the things that you control (your data) and the things that you do not control (the Internet). Typically, this is the router used to separate your network from your Internet Service Provider (ISP). In this area, you exchange information with limited, calculated risk. Creating a DMZ involves adding firewalls for extra layers of security. Firewalls are often used in multi-machine systems to protect the resources that live on that private network, such as critical data, business applications, and sensitive information. A wide variety of topographies can be appropriate for a DMZ; however, the basic units usually look something like the layout in Figure 2-7.

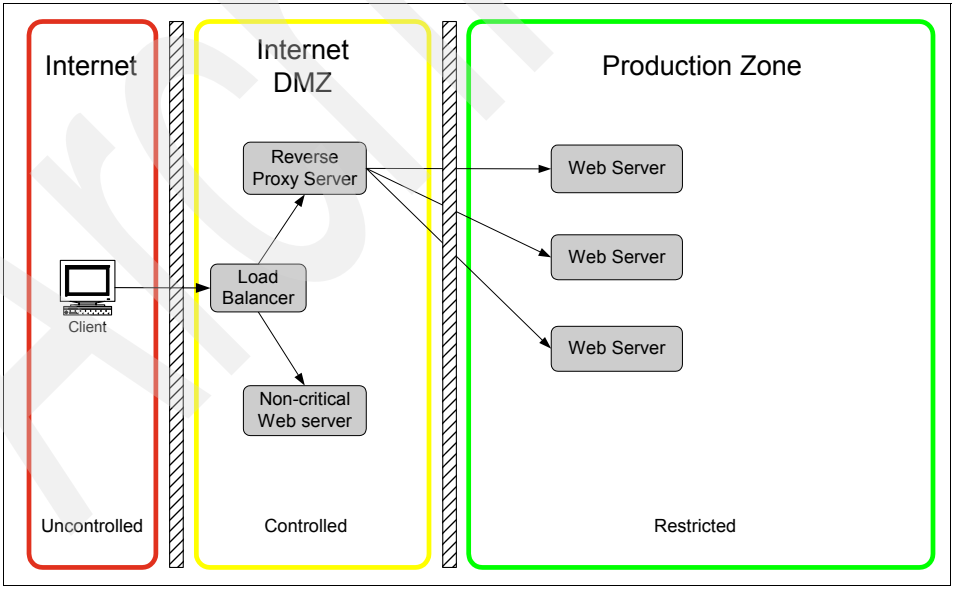


Figure 2-7 Basic DMZ design

This design allows for the separation of the presentation material on the non-critical Web server and the application logic on the Web servers in the private network. The infrastructure allows secure transactions and processing in stages, reducing the demands on systems

Most firewalls and security schemes are built to keep the Internet away from the internal network. However, in some situations, you may want to protect parts of the internal network from other areas of your internal network. It makes sense that not everyone needs access to the same services, information, or security protection. Figure 2-8 shows the segregation of the intranet client from the production environment. Some parts of your enterprise need to be more secure than others, such as demonstration networks (where there are often people from outside of the organization present), Human Resources data, development projects, financial data, and so on.

Adding the additional security of another reverse proxy to the network gives you manageability of the internal user's access as well. In this example, the user has been allowed full access to one Web server (solid line), limited access to one other Web server (broken line), and no access to the remaining server.

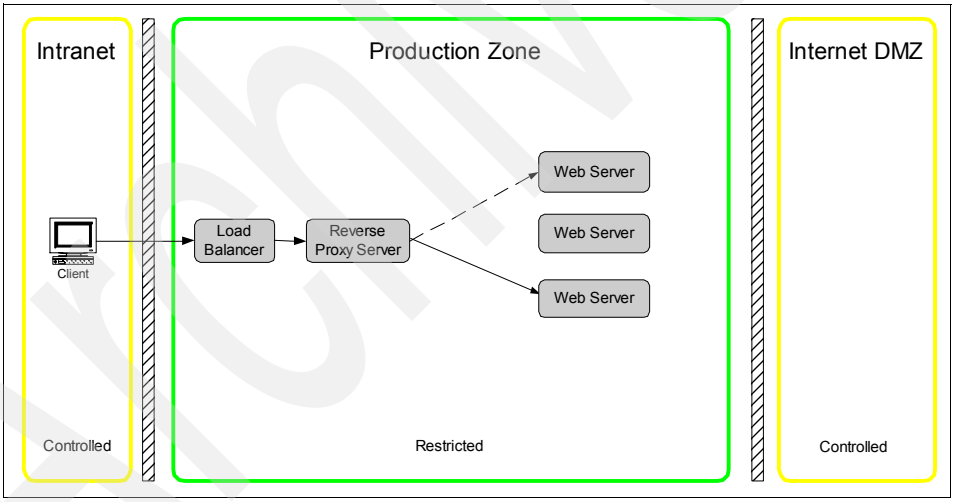


Figure 2-8 Segregating the intranet client

Let us take that concept one step further. in Figure 2-9 on page 39, we add an additional zone of protection and tie the idea of load balancing, as well as high availability, into the architecture. By moving the security management into its own area that is physically and virtually secure, you create an area where the security administration will be performed, and all of the necessary data is contained only in that area.

You can undertake this type of segregation of the network for various reasons. You could create another protected area called Human Resources, where the applications and data would all be contained inside that specific network with access granted only as needed. Take care when applying this type of result. Separate the things that absolutely must be protected. Keep your solution straightforward and easily scalable for future growth.

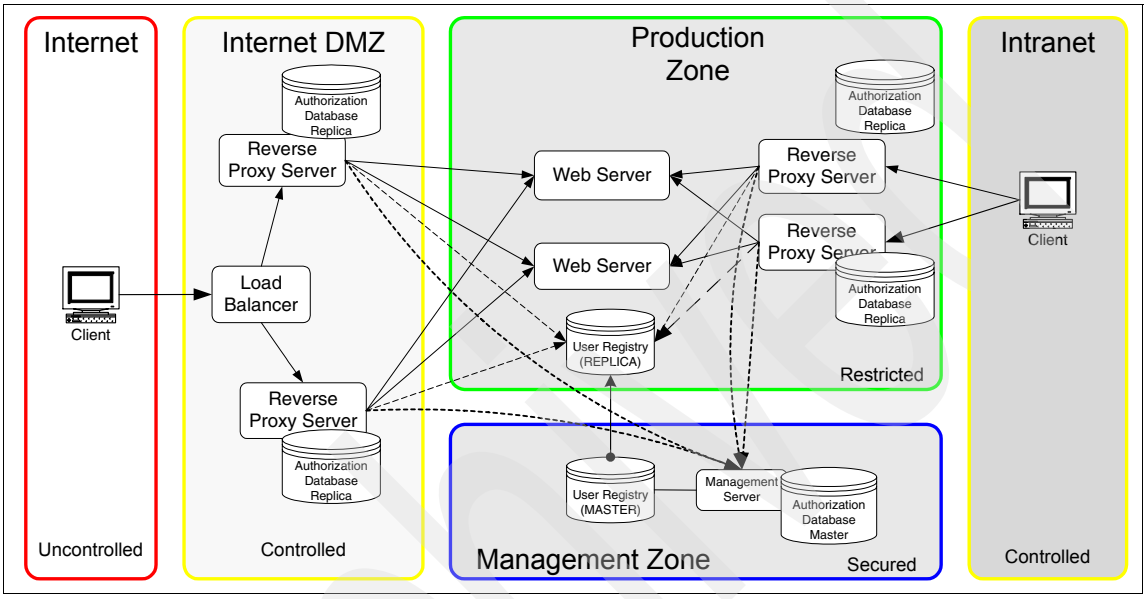


Figure 2-9 Management zone, high availability, and load balancing

2.5 Additional components

The discussion on practical designs in 2.4, “Practical designs” on page 37, introduces system components into our discussion. There are several components we will find through the remainder of this book.

Web server

A *Web server* is simply a server that processes http and https requests. The requests may be for content stored on, or developed on the server itself, or the Web server may present responses produced after the Web server has passed requests to other systems.

Web application servers

A *Web application server*, or application server is a server that is running an application. This application may be coupled with a Web server on the same

system, or receive requests from a standalone Web server. The application server and Web server, when separate, may exist in separate network zones.

Portal

A *portal* represents a way to provide access to a variety of applications from one Web location. The portal represents a single location to the user, making the transition to the various locations seamless and transparent.

Backend

The *backend* portion refers to the part of the system that actually processes the requests and provides information. This often includes database and mainframe systems that are used.

Database

A *database* is a collection of data stored for use by applications. The data may or may not be related.

Messaging services

The *messaging service*, and messages can take many forms. At the basic level a message is data sent between two devices. This could take several forms, from Web services, to e-mail, to text messages in a wireless environment, to name a few. Messaging services deal with the transport and delivery of these messages. For our purposes the messaging service enables communications between devices.

Service-oriented architecture

A service-oriented architecture (SOA) reflects distributed services that communicate with each other used to meet requirements. These services are *orchestrated* to process data and data requests. Each service operates independently with it's own state and context. Each service has a clearly defined method to use. Most service-oriented architectures involve Web services using *SOAP* and *WSDL*. SOAP is a method for exchanging XML messages over the http protocol. WSDL stands for Web Services Description Language, an XML method for describing the available Web services.

2.6 Access control models

Our security architecture design requires selection of a model for access control. This model must match our security policy regarding how to manage access to resources. Our selection of a model is influenced by regulations, and also by the type of resource protection we choose. There are three main models, Role

Based Access Control, discretionary access control, and mandatory access control.

Role Based Access Control

Role Based Access Control (RBAC), as its name suggests, is the granting of access privileges to a user based upon the work they perform within an organization. A user can be assigned to a single or multiple roles either automatically or manually. The membership to each role grants access to specific resources.

Discretionary Access Control

Within the DAC model the owner of a resource decides on whether to allow a specific person access to their resource. This system is common in distributed environments that have evolved from smaller operations into larger ones. When it is well managed, it can provide adequate access control, but it is very dependant upon the resource owner understanding how to implement the security policies of the organization, and of all the models, it is most like to be subject to *management by mood*. Ensuring that authorized people have access to the correct resource requires a good system for the tracking of leavers, joiners, and job changes. Tracking requests for change is often paper driven, error prone, and can be costly to maintain and audit.

Mandatory access control

The mandatory access control (MAC) model is where the resources are grouped and marked according to a sensitivity model. This model is most commonly found in military or government environments. One example would be the markings of Unclassified, Restricted, Confidential, Secret, and Top Secret. Users privileges to view certain resources will be dependant upon that individuals clearance level.

2.6.1 Which model

All three models previously discussed have pros and cons associated with them. Which model an organization uses will depend upon a number of factors, including, but not limited to, externally mandated policies, maturity of existing identity management processes, range of identity management target systems, future requirements, number of users managed, and risk assessment and return on investment statistics.

RBAC

The key to this model is the ability to classify users by what resources they should be allowed to access. The following examples indicate some roles, and the resources for each.

A new customer Alex registers with an organization by completing a form on a Web site. As a result of doing so, Alex may be awarded the role of “customer” by the central user administration system that in turn populates Alex's account to all customer-facing resources.

A new employee Betty, on starting with an organization, could be awarded the role of “basic user” by the administrator and as a result, her account information could be populated to the network access system and to an e-mail system. Betty may not yet have interacted with any of the systems, so in this case, the administrator would have to assign the accounts with a default password and ensure that each system makes Betty change her password upon first access.

A senior employee Charles would already have the “basic user” role from the time when he joined the organization. His work now requires that access be granted to applications that are not included within the “basic user” role. If he now needs access to the accounts and invoicing systems, Charles could be awarded the “accounting” role in addition to the “basic user” role.

A manager Dolly would already have the “basic user” role from the time when she joined the organization and may also have other roles. As she has been promoted to a management post, so her needs to access other systems have increased. It may also be, however, that her needs to access some systems, as a result of her previous post, are no longer appropriate in her management role. Thus if Dolly had “basic user” and “accounting” as her roles before promotion, it may be that she is granted the “manager”, but has her “accounting” role rescinded. This would leave her with the “basic user” and “manager” roles suitable for her post.

MAC

The key to this kind of system is the ability to use background security checking of personnel to a greater level than that which would normally be carried out in a business or non-governmental environment. It is also key for data of different

sensitivity to be kept segregated. For example, a user must not be able to cut and paste information between documents of differing sensitivities. This has traditionally been achieved by keeping data physically separate. In this environment, therefore, a user may have a number of different workstations; one for restricted, one for secret, and so on, each running on completely different and separate architectures.

Conducting identity management across multiple sensitivity silos with one central identity management system raises a number of issues. The central system itself must be classified at the highest level, as it holds user rights to all sensitivity silos. Normally in this environment, this would mandate that various security certifications and accreditation processes have been completed and also that any cryptographic keys are in hardware storage.

As the Web portal approach matures, this kind of multiple silo approach may change, but in the short term, this would mean that a software only solution would not be possible.

One further approach would be to treat each sensitivity silo as a discrete identity management problem. This would mean that there is a distinct solution for each silo and that the best access control model could be chosen from the other two. For example, at the lowest sensitivity silo, there are likely to be many more users that best fit an RBAC solution, while at the top level, there are fewer users and other (physical, procedural, personnel, and technical) more rigorous controls, so a DAC might be more appropriate.

Despite its limitations, this type of access control model will continue to be used in military and government environments, because it provides the solid foundation for segregation of information based upon sensitivity. Identity management solutions for this space are probably best focused on the lower sensitivity silo, unless approvals can be gained to connect all silos with a highly secure management layer that includes identity management.

DAC

Discretionary Access Control is the model that is most likely to be used as a default or evolved decentralized access control solution. Organizations are familiar with the concept of each application administrator or owner being responsible for granting access to the application or system owned or administered by them. Key features of a centralized identity management system that allows this to continue are the ability to specify over-arching corporate security policies, combined with the ability to delegate responsibility for account management to individual systems. A centralized identity management system with these features allows for a reduction in the amount of “management by mood”, but ensures that corporate security policies can be applied, while allowing a degree of actual and real political ownership of the target resource.

The different access control models are compared in Table 2-2.

Table 2-2 Access control model comparison and notes on desirable features

Access control model	Pros	Cons
MAC	1. Ideally suited to military and government security requirements. 2. Highly secure.	1. Costly to implement because of personnel vetting and data segregation requirements. 2. Difficult to centrally manage all identities because of sensitivity silos.
DAC	1. Likely to already be in use. 2. Easy to implement centralized identity management solution. 3. Suited to most commercial organizations, prior to centralized identity management or during conversion to RBAC.	1. Subject to management by mood. 2. Policy enforcement and audit costly. 3. Centralized identity management possible but less return on investment (ROI) than single RBAC model.
RBAC	1. Useful for strong role focused organizations. 2. Useful for organizations with high staff turnover and reliance on temporary or casual staff. 3. Recommended for large user populations, particularly where users include customers and partner organizations.	1. RBAC design can be difficult politically and logically. 2. Strong policies required particularly where delegated administration is used.

2.7 Certificates

A digital signature is a way to ensure that an electronic document or communication is authentic. Digital certificates contains information needed to verify the digital signature. Digital signatures rely on certain types of encryption to ensure authentication. *Encryption* is the process of encoding all of the data that one computer sends to another in a form that only the other computer will be able to decode. *Authentication* is the process of verifying that information comes from a trusted source.

There are several ways to authenticate a person or information about a computer. Two of the most frequently employed are:

- Private key encryption

  With private key encryption, each computer has a secret key (code) that it can use to encrypt a packet of information before it is sent over the network to the other computer. This mechanism requires that you know which computers will talk to each other in order to distribute and install the key on each one. Private key encryption is essentially the same as a secret code that the two computers must each know in order to decode the information.

- Public key encryption

  Public key encryption uses a combination of a private key and a public key. The private key is known only to your computer, but the public key is given to any computer that wants to communicate securely with it.

  A sender uses your public key to encrypt a message before he sends it to you. To decode this encrypted message, your computer uses your private key. This way the message is protected while in transit, nobody is able to decode it but you.

  Public key encryption is a technique that uses a pair of asymmetric keys for encryption and decryption. Each pair of keys consists of a public key and a private key. The public key is made public by distributing it widely. The private key is never distributed; it is always kept secret. Data that is encrypted with the public key can be decrypted only with the private key.

  This technique is also being used for signing in order to prove the origin of data. A message can be signed using the private key of the sender, and anyone who receives the message can use the sender's public key to verify the origin of the message. This asymmetry is the property that makes public key cryptography so useful.

Digital certificates are used for Secure Sockets Layer (SSL) technology, the industry-standard method for protecting Web communications developed by Netscape Communications Corporation. The SSL security protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. Because SSL is built into all major Web browsers and servers, simply installing a digital certificate turns on the browser's SSL capabilities.

The certificate represents a credential issued by a certificate authority, testifying to the authenticity of the identified party. The certificate contains the identified party's public key, so that the recipient of messages from the party can use it to decrypt messages encrypted using the party's private key.

Transmitting sensitive data, such as credit card numbers and health care data, across the Web and intranets requires authentication to ensure that the destination of the data is legitimate, encryption to protect the data against interception or tampering, and *message integrity* to ensure that the information is not tampered with during transmission. SSL is the standard technology used to protect information transmitted over the Web via HTTP protocol and protects against site spoofing, data interception, and tampering.

Protecting the confidentiality and integrity of sensitive information transmitted over the network is a crucial step to building customer confidence, securely interacting with business partners and complying with new privacy regulations. Because of the increasing awareness and concerns regarding confidentiality and data integrity, the exchange of information between Web servers and clients, server-to-server, and among other networking devices, must be protected with cross-network security mechanisms for servers facing both the Internet and private intranets.

Certificates, which are based on the open standard X.509, contain this information:

- Version number (certificate format)
- Serial number (unique value from CA)
- Algorithm ID (signing algorithm used)
- Issuer (name of CA)
- Period of validity (from and to)
- Subject (user's name)
- Public key (user's public key & name of algorithm)
- Digital signature
- Created by CA
- Encrypted with CA's private key

Managing certificates can be arduous. You can install your own Public Key Infrastructure (PKI) and maintain it, or use the services of Certificate Authorities (CAs), the digital world's equivalent of passport offices. CAs issue digital certificates and validate the holder's identity and authority. They embed an individual's or an organization's public key along with other identifying information into each digital certificate and then cryptographically "sign" it as a tamper-proof seal, verifying the integrity of the data within it and validating its use.

For more information about certificates visit:

`http://developer.netscape.com/tech/security/ssl/howitworks.html`

For more information about Certificate Authorities:

`http://www.verisign.com/`
`http://www.thawte.com/`

2.8 Security components

There are common components used to provide the functionality of the security architecture subsystems. A reminder that one component may provide the function for a subsystem, or that several components may be required. Also the components may be involved in several of the subsystems. For example, an identity management component providing the identity or credential subsystem may send information to the security audit subsystem. It participates in both. This type of participation is evident in Figure 2-1 on page 22.

The remainder of this book deals with using IBM Tivoli Security Solutions to provide functionality for the five security architecture subsystems. The specific products involved are:

Table 2-3 IBM Tivoli Security Solutions

Product	Product level
IBM Tivoli Identity Manager	V 4.6
IBM Tivoli Identity Manager Express	V 4.6
IBM Tivoli Access Manager for e-business	V 6.0
IBM Tivoli Access Manager for Operating Systems	V 6.0
IBM Tivoli Access Manager for Business Integration	V 5.1
IBM Tivoli Access Manager for Enterprise Single Sign-On	V 6.0
IBM Tivoli Directory Integrator	V 6.1.1
IBM Tivoli Directory Server	V 6.0
IBM Tivoli Security Compliance Manager	V 5.1.1
IBM Tivoli Security Operations Manager	V 3.1
IBM Tivoli Common Audit and Reporting Service	V 6.0
IBM Tivoli Federated Identity Manager	V 6.1
IBM Tivoli Federated Identity Manager Business Gateway	V 6.1

In the remainder of the book we use these products as components for our security architecture subsystems, utilizing common network models to develop a security architecture. This is based on the foundation of determining the business context, assessment of risk and business drivers, and developing an appropriate security policy. The components and processes relating to their implementation will be discussed as well.

2.9 Conclusions

We have seen that each security architecture can be different, based upon the business context (Chapter 1, “Business context” on page 3), and yet at the same time there is commonality in approaching their design. We can use the common elements as a starting point for our security architecture. The choice, location, and configuration of technologies and products, along with processes, allows customization to meet the individual business context.

Building a secure system is not enough. Keeping it functional, testing it, and improving and reviewing it with management and your security, network, and development professionals is mandatory. When you deal with the Web and network security, reviewing your procedures and policies regularly helps keep the enterprise protected from new threats as well as old.

A question to keep in mind as you review your environment: “Will the cost of this improvement be more or less than repairing or replacing the assets compromised or lost?”

It is generally more cost effective to be proactive rather than reactive.

In the following chapter we take a closer look at some of the foundation technology needed for all security architecture implementations—directories and directory integration.

3

Directory technologies

In Chapter 2, “Common security architecture and network models” on page 19, we introduced five different subsystems that address access control, identity and credentials, flow control, integrity, and audit that can be used to design a security architecture. Every subsystem provides unique functionality that can solve specific tasks. Alternatively, there are infrastructure elements that are needed to provide cross-subsystem services. A directory is one of these components that cannot be mapped into one distinct category but offers a broad spectrum of capabilities. This chapter addresses these capabilities in five distinct parts.

In the first part we explain why an organization should use a centralized directory server as its user repository. We emphasize the need to consolidate the definition of all of the users who have access to any resource in one or at most a few repositories.

In the second part we introduce the concept of directory and LDAP. We show the main features of LDAP based directory servers focusing on the architectural and security point of view. The content of this section is independent of IBM-specific implementations of directory servers.

In the third section we show the directory server developed by IBM: IBM Tivoli Directory Server.

In the forth part we compare two directory integration technologies, meta directories and virtual directories.

In the last section we show the directory integration product offered by IBM: IBM Tivoli Directory Integrator.

3.1 Using a centralized user repository

Increasingly, enterprises are seeking to improve operational efficiencies and expand their businesses by opening their internal systems to a broader community of their systems, employees, customers, and suppliers. A consistent and reliable identity infrastructure enables enterprises to expose their internal processes to their supply chain, their customers, and the growing mass of automated machine-to-machine transactions. A common user repository is a key enabler for security and application infrastructure in an enterprise.

In the first two parts of this section we introduce the business and technical requirements for a centralized repository.

A centralized directory server can address these requirements even if some practical considerations are necessary regarding using one centralized or multiple repositories, which we discuss in the third section.

Finally we discuss why a directory server is the right choice as a user repository with respect to other technologies.

3.1.1 Business requirements

In this section we show a brief summary of the business drivers involved with a consistent identity infrastructure. Refer to Part 3, “Managing identities and credentials” on page 507 for a broader analysis of the issues related to this topic.

A centralized repository is meant to consolidate all user definitions into only one data source. Most companies, while expanding their business, increase the number of applications and platforms, each with its own format and place for defining the enabled users. The final result is that user credentials are stored in a number of different and disjointed places. This means that the same person might have different, and not synchronized, accounts for different applications. In large companies the number of these accounts may reach double-digit or even triple-digit numbers. The main problems include:

- High costs for user management. Expenses increase proportional to the number of repositories. Included in these costs:
  - User additions, modifications, and deletions have to be repeated in each repository.
  - Password management is one of the highest costs for a company’s help desk.

  - Training costs. Administrators have to be skilled on different products and platforms.
  - Costs in terms of hardware resources and system administrators.
- Security
  - Users have many passwords and do not protect them properly.
  - Policies cannot be enforced consistently across the business.
  - Effort to protect data spread in various locations.
  - Longer time to deny a person access to any company's data.
- Data integrity. Information could be inconsistent across the business.
- Higher risks related to human errors, malicious attacks, and system failures.
- Availability and scalability of the systems.

The common problems outlined above can be faced and mostly solved by consolidating the disjointed data sources in only one manageable, available, and scalable repository. This is one of the basic concepts of implementing centralized security, provisioning, and Web services. It also helps define an authoritative source of user identities and to establish clear and uniform processes to manage user definitions.

3.1.2 Functional requirements

The business requirements introduced above turn into the following functional requirements:

- Have all of the applications and operating systems share the same user definitions. This implies that there is a single point of administration for all of the company's user accounts.
- Have identity information consistent across all repositories. For example, this is important to secure user passwords. If users have different passwords for every application and platform, they end up adopting easy passwords or they record them and do not secure them properly.
- Enforce the same security policies across different applications and operating systems.
- Avoid storing redundant data in different locations.
- Implement a small number of servers or just one highly available and easily scalable architecture, reducing the number of clusters and replicas of data.

In order to satisfy these requirements, more than one approach is possible. One of the most intuitive solutions is to consolidate all user definitions in one repository and modify applications and operating systems to utilize this central

repository. However, this operation might be very difficult, therefore more than one repository might be necessary.

3.1.3 One or multiple repositories

Large companies develop their IT environments over many years without paying close attention to sharing user credentials across the organization. For them it could be extremely expensive to adopt a centralized user repository that provided user identities to all existing applications and operating systems. In fact, it would require modifying applications and operating system configurations, and, in some circumstances, developing new code as well. In addition, for some applications it would be too expensive or even impossible to perform authentication tasks against the central repository.

It is good practice to have different applications use authentication mechanisms that utilize a common repository. This should be implemented for new applications, and existing applications may be modified as well. But even after this consolidation, there could still be repositories in a large environment that have to be maintained. There is no rule for the number of repositories that are acceptable, but it is a good idea to reduce the number as much as possible. However, costs and technical or user management issues could limit the consolidation. For example, a common technical reason is the difficulty in integrating some applications with the designed centralized repository. A user management reason is to avoid additional education for administrators on a new product, but to keep them working with tools and utilities they already know and that are well-customized for the specific application.

Consolidating user credentials in a few repositories rather than in one might simplify the adaptations necessary on applications and operating systems. Nevertheless, in order to achieve a consistent identity infrastructure it is necessary to integrate these repositories. This means that the effort is focused in a different direction. In Part 3, “Managing identities and credentials” on page 507, we introduce two solutions to integrate different data sources, one based on IBM Tivoli Directory Integrator and the other based on IBM Tivoli Identity Manager.

A common scenario of a company that adopts different platforms and decides to maintain multiple user repositories might include a Human Resources database, an IBM Tivoli Directory Server, a Microsoft® Active Directory®, and an IBM Lotus® Domino® environment. This scenario is not unusual and it is interesting to note that three out of four repositories can be regarded as directory servers. In the remainder of this section we explain what directories are and how they work.

3.1.4 Why a directory server

For our discussion we assume that the main reason for using a directory server as a user repository is because it is a standard. This means that most applications, operating systems, and middleware products of many vendors come with LDAP support, where LDAP is a common protocol to access directories. Therefore, access is allowed or denied by verifying user credentials stored in a directory. The standardization of the access protocol is the key to having a centralized company directory.

We also need to consider the reasons why directories and LDAP became standard. Basically their structure and features are optimized for the purpose of a user repository. The next sections are dedicated to explaining directory server principles and to clarifying these reasons.

3.2 Directories

In this section we introduce the concepts of directory and LDAP. Then we describe the main features of directory servers, focusing on architecture and security. In particular, we describe methods for securing data within a directory and to control access to them. Then we see how to organize data within a directory and how to build a secure, scalable, and highly available physical architecture integrating LDAP servers in a company network. Finally, we show how to perform administrative tasks.

3.2.1 General definition

A directory is a listing of information about objects arranged in some order and providing details about each object. Common examples are a city telephone directory and a library card catalog. For a telephone directory, the objects listed are people; the names are arranged alphabetically, and the details given about each person are address and telephone number. Books in a library card catalog are ordered by author or by title, and information such as the ISBN attribute of the book and other publication information is given.

In computer terms, a directory is a specialized database, also called a data repository, that stores typed and ordered information about objects. A particular directory might list information about printers (the objects) consisting of typed information such as location (a formatted character string), speed in pages per minute (numeric), print streams supported (PostScript®, ASCII), and so on.

Directories enable users and applications to find resources that have the characteristics needed for a particular task. For example, a directory of users can be used to look up a person's e-mail address or fax number. A directory could be

searched to find a nearby PostScript color printer. A directory of application servers could be searched to find a server that can access customer billing information.

The terms *white pages* and *yellow pages* are particular directory applications. If the name of an object (person, printer) is known, its characteristics (phone number, pages per minute) can be retrieved. This is similar to looking up a name in the white pages of a telephone directory. Or, if the name of a particular individual attribute is not known, the directory can be searched for a list of objects that meet a certain requirement. This is like looking up a listing of hairdressers in the yellow pages of a telephone directory. However, directories stored on a computer are much more flexible than the yellow pages of a telephone directory, because they can usually be searched by a range of criteria, not just by a single predefined set of categories.

3.2.2 Directory versus database

A directory is often described as a database, but it is a specialized database that has characteristics that set it apart from, for example, general-purpose relational databases. One special characteristic of directories is that in general they are accessed (read or searched) much more often than they are updated (written). Hundreds of people might look up an individual's phone number, or thousands of print clients might look up the characteristics of a particular printer. But the phone number or printer characteristics seldom change.

Directories must be able to support high volumes of read requests, so they are typically optimized for read access. Write access might be limited to system administrators or to the owner of each piece of information. A general-purpose database, on the other hand, needs to support applications such as airline reservations and banking with high update volumes. Directories are not appropriate for storing information that changes rapidly and frequently. For example, the number of jobs currently in a print queue probably should not be stored in the directory entry for a printer because that information would have to be updated frequently to be accurate. Instead, the directory entry for the printer could contain the network address of a print server. The print server could be queried to learn the current queue length if desired. The information in the directory (the print server address) is static, while the number of jobs in the print queue is dynamic.

Another important difference between directories and general-purpose databases is that directories may not support transactions, although IBM Tivoli Directory Server does. Transactions are all-or-nothing operations that must be completed in total or not at all. For example, when transferring money from one bank account to another, the money must be debited from one account and credited to the other account in a single transaction. If only half of this transaction

completes or someone accesses the accounts while the money is in transit, the accounts will not balance. General-purpose databases usually support such transactions, which complicates their implementation. Because directories deal mostly with read requests, the complexities of transactions can be avoided. For example, if two people exchange offices, both of their directory entries must be updated with new phone numbers, office locations, and so on. It is considered acceptable if one directory entry is updated first, and then other directory entry is updated later, so allowing a brief period during which the directory will show that both people have the same phone number.

In contrast to directories, general-purpose databases must support arbitrary applications such as banking and inventory control, so they allow arbitrary collections of data to be stored. On the other hand directories may be limited in the type of data they allow to be stored (although the architecture does not impose such a limitation). For example, a directory specialized for customer contact information might be limited to storing only personal information such as names, addresses, and phone numbers. If a directory is extensible, it can be configured to store a variety of types of information, making it more useful to a variety of programs.

Another important difference between a directory and a general-purpose database is in the way information can be accessed. Most databases support a standardized, very powerful access method called Structured Query Language (SQL). SQL allows complex update and query functions at the cost of program size and application complexity. Directories, on the other hand, use a simplified and optimized access protocol that can be used in slim and relatively simple applications.

In the following section we introduce the most common protocol to access directories: Lightweight Directory Access Protocol (LDAP).

3.2.3 LDAP: Protocol or directory

LDAP defines a communication protocol. That is, it defines the format of messages used by a client to access data in a directory service that listens for and responds to LDAP requests. LDAP does not define the directory service itself, yet people often talk about LDAP directories. Others say LDAP is only a protocol, that there is no such thing as an LDAP directory. What is an LDAP directory?

LDAP evolved as a lightweight protocol for accessing information in X.500 directory services. It has since become independent of X.500 and now is the standard protocol to access directories. Directory servers that specifically support the LDAP protocol rather than the X.500 Directory Access Protocol (DAP) generally are called LDAP servers.

The success of LDAP has been largely due to the following characteristics that make it simpler to implement and use, compared to X.500 and DAP:

- LDAP runs over TCP/IP rather than the OSI protocol stack. TCP/IP is less resource-intensive and is much more widely available, especially on desktop systems.
- The functional model of LDAP is simpler. It omits duplicate, rarely used and esoteric features. This makes LDAP easier to understand and to implement.
- LDAP uses strings to represent data rather than complicated structured syntaxes such as ASN.1 (Abstract Syntax Notation One).
- LDAP provides an API (application programming interface) that enables applications to interact easily with LDAP servers. The API can be considered an extension to the LDAP architecture.

Refer to the IBM Redbooks *Understanding LDAP - Design and Implementation*, SG24-4986, for more details about the LDAP protocol and related RFCs. In this book, the term LDAP refers to LDAP Version 3.

3.2.4 DSML

Recently, the push for encapsulating LDAP operations within XML for use within Web services has spawned a new language called the Directory Services Markup Language (DSML). The most recent of the specification is DSMLv2. DSML is an XML schema for representing directory information, it is a generic import / export format for directory information. Directory information in DSML can be shared between DSML-aware applications without exposing the LDAP protocol.

XML provides an effective way to present and transfer data; Directory services allow you to share and manage data, and are thus a necessary prerequisite for conducting online business; DSML is designed to make directory services more dynamic by employing XML. DSML is an XML schema for working with directories, it is defined using a Document Content Description (DCD). Thus, DSML allows XML programmers to access LDAP-enabled directories without having to write to the LDAP interface or use proprietary directory-access APIs, and provides one consistent way to work with multiple dissimilar directories.

3.2.5 Directory clients and servers

Directories are usually accessed using the client/server model of communication. An application that wants to read or write information in a directory does not access the directory directly. Instead, it calls a function or application programming interface (API) that causes a message to be sent to another process. This second process accesses the information in the directory on behalf

of the requesting application via TCP/IP. The default TCP/IP ports are 636 for secure communications and 389 for unencrypted communications. The results of the read or write action are then returned to the requesting application, as shown in Figure 3-1.

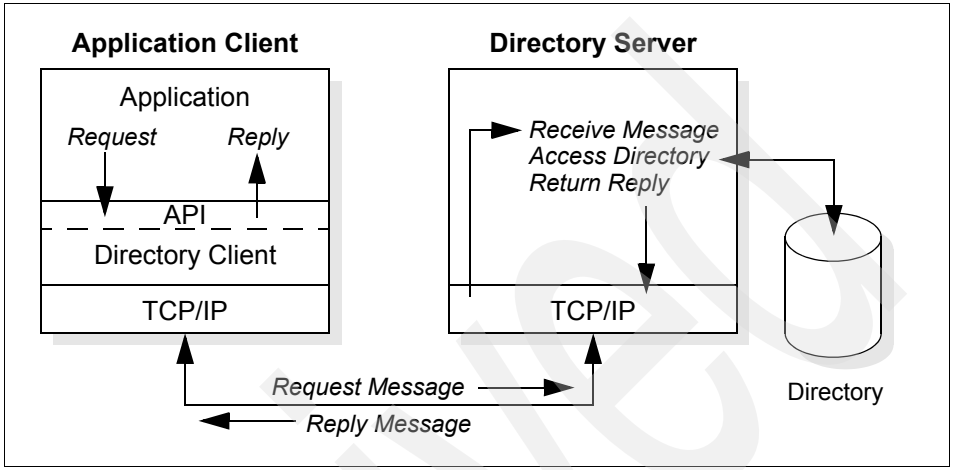


Figure 3-1 Directory client/server interaction

The request is performed by the directory client, and the process that maintains and looks up information in the directory is called the directory server. In general, servers provide a specific service to clients. Sometimes, a server might become the client of other servers in order to gather the information necessary to process a request.

The client and server processes might or might not be on the same machine. A server is capable of serving many clients. Some servers can process client requests in parallel. Other servers queue incoming client requests for serial processing if they are currently busy processing another client's request.

An API defines the programming interface that a particular programming language uses to access a service. The format and contents of the messages exchanged between client and server must adhere to an agreed-upon protocol. LDAP defines a message protocol used by directory clients and directory servers. There are also associated LDAP APIs for C and Java™ languages, and ways to access the directory from a Java application using Java Naming and Directory Interface™ (JNDI). The client is not dependent on a particular implementation of the server, and the server can implement the directory however it chooses.

3.2.6 Distributed directories

The terms *local, global, centralized,* and *distributed* are often used to describe a directory. These terms mean different things in different contexts. In this section, we explain how these terms apply to directories.

In general, *local* means nearby, and *global* means that something is spread across the universe of interest. The universe of interest might be a company, a country, or the Earth. Local and global are two ends of a continuum. That is, something may be more or less global or local than something else. *Centralized* means that something is in one place, and *distributed* means that something is in more than one place. As with local and global, something can be distributed to a greater or lesser extent.

The information stored in a directory can be simultaneously local and global in scope. For example, a directory that stores local information might consist of the names, e-mail addresses and so on of members of a department or workgroup. A directory that stores global information might store information for an entire company. Here, the universe of interest is the company.

The clients that access information in the directory can be local or remote. Local clients may all be located in the same building or on the same LAN. Remote clients might be distributed across the continent or planet.

The directory itself can be *centralized* or *distributed*. If a directory is centralized, there may be one directory server at one location or a directory server that hosts data from distributed systems. If the directory is distributed, there are multiple servers, usually geographically dispersed, that provide access to the directory.

When a directory is distributed, the information stored in the directory can be *partitioned* or *replicated*. When information is partitioned, each directory server stores a unique and non-overlapping subset of the information. That is, each directory entry is stored by one and only one server. One of the techniques to partition the directory is to use LDAP referrals. LDAP referrals enable users to refer LDAP requests to a different server. When information is replicated, the same directory entry is stored by more than one server. In a distributed directory, some information may be partitioned while some may be replicated.

The three *dimensions* of a directory—scope of information, location of clients, and distribution of servers—are independent of each other. For example, clients scattered across the globe can access a directory containing only information about a single department, and that directory can be replicated at many directory servers. Or, clients in a single location can access a directory containing information about everybody in the world that is stored by a single directory server.

The scope of information to be stored in a directory is often given as an application requirement. The distribution of directory servers and the way in which data is partitioned or replicated often can be controlled to affect the performance and availability of the directory. More details about this topic are shown in 3.2.10, “Availability and scalability” on page 68.

3.2.7 Directory security

The security of information stored in a directory is a major consideration. Directories are used for different scopes, both for Internet and intranet users, but in all cases any user should not necessarily be able to perform any operation. Directories should be placed in restricted access zones, but security control must be performed by the directory server itself.

For example, any intranet user should be able to look up an employee’s e-mail address, but only the employee themselves or a system administrator should be able to change it. Members of the personnel department might have permission to look up an employee’s home telephone number, but their co-workers might not. Depending on the confidentiality of the data, information may have to be encrypted before being transmitted over the network. A security policy defines who has what type of access to what information, and is defined by the organization that maintains the directory.

A directory should support the basic capabilities needed to implement a security policy. The directory in this case is one of the components by which security is provided to the whole network. It is also one of the network resources that needs to be protected.

Directory security covers the following four aspects:

- Authentication
- Integrity
- Confidentiality
- Authorization

Authentication

Authentication is the verification of the identity claimed by the requester (machine or person). This can be realized in several methods:

No authentication This is the simplest authentication method, one that obviously does not need to be explained in much detail. This method should only be used when data security is not an issue and when no special access control permissions are involved. This could be the case, for example, when your directory is an address book browsable by anybody. No authentication is assumed

when you leave the password and DN fields empty in an LDAP operation. The LDAP server then automatically assumes an anonymous user session and grants access with the appropriate access controls defined for this kind of access (not to be confused with the SASL anonymous user).

Basic Authentication (BA)

When using basic authentication with LDAP, the client identifies itself to the server by means of a DN and a password which are sent in the clear over the network (some implementations may use Base64 encoding instead). The server considers the client authenticated if the DN and password sent by the client match the password for that DN stored in the directory. Base64 encoding is defined in the Multipurpose Internet Mail Extensions (MIME) LDAP standard (RFC 1521). It is a relatively simple encryption, and therefore it is not hard to break once one has captured the data in the network.

Simple Authentication and Security Layer (SASL)

SASL is a framework for adding additional authentication mechanisms to connection-oriented protocols. It has been added to LDAP Version 3 to overcome the authentication shortcomings of Version 2. It is a proposed Internet standard defined in RFC 2222.

In SASL, connection protocols, like LDAP, IMAP, and so on, are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. A complete list of SASL profiles can be obtained from the Information Sciences Institute[1] (ISI). Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. Optionally, a security layer can be negotiated to encrypt the data after authentication and so ensure confidentiality. LDAP Version 3 includes a command (ldap_sasl_bind()) to encrypt the data after authentication.

SSL/TLS

SSL/TLS supports server authentication (client authenticates server), client authentication (server authenticates client), or mutual authentication. In addition, it provides for privacy by encrypting data sent over the

[1] Refer tothe following Web site for more information:
http://www.isi.edu/

network. SSL/TLS uses a public key method to secure the communication and to authenticate the counterparts of the session. This is achieved with a public/private key pair. They operate as reverse functions to each other, which means data encrypted with the private key can be decrypted with the public key and vice versa. The assumption for the following considerations is that the server has its key pair already generated. This is usually done when setting up the LDAP server.

Integrity

Integrity is the assurance that the information that arrives is really the same as what was sent.

Confidentiality

Confidentiality is assuring, through data encryption, that information reaches only those for whom it is intended. For example, sensitive data such as passwords can be stored encrypted in the directory, or network transmissions can be protected using SSL. See 2.7, “Certificates” on page 44 for more about SSL.

Authorization

Authorization is the verification that someone is really allowed to do what he is requesting to do. This is usually checked after user authentication by verifying ACLs. An ACL is a list of authorizations such as read, write, and delete that is given to a subject who may be attached to objects and attributes in the directory. An ACL lists the type of access to an object that each user or a group of users is allowed or denied. In order to make ACLs shorter and more manageable, users with the same access rights are often put into security groups. Table 3-1 shows an example ACL for an employee’s directory entry.

Table 3-1 Example ACL for an employee’s directory entry

User or group	Access rights
owner	read, modify (but not delete)
administrators	all
personnel	read all fields
all others	read restricted

In the following section we explain how to organize data within a directory.

3.2.8 Schema and namespace

Structuring data is done by designing a schema, choosing a directory suffix, branching the directory tree and, finally, creating a naming style for the directory entries. We explain these activities in the sections that follow.

Directory schema

A directory entry usually describes an object such as a person, a printer, a server, and so on. Information is stored into entries that are described by attributes. An object class consists of a set of mandatory and optional attributes. A schema is the collection of attribute-type definitions and object class definitions. Every entry in the directory has an object class associated with it. Thus, every entry in the directory contains a set of mandatory and optional attributes based on the entry's object class and that object class definition. Attributes are typed in the form of <type>=<value> pairs in which the type is defined by an object identifier (OID) and the value has a defined syntax. Attributes can be single-valued or multi-valued. The allowed set of characters for object and attribute names is defined in the Attribute Syntax Definitions RFCs of LDAP protocol (v3). For more details about LDAP Version 3 RFCs refer to the IBM Redbook *Understanding LDAP - Design and Implementation*, SG24-4986.

When deciding on the design of the schema, there are a few things to consider. LDAP specifications provide a standard schema for a broad range of applications including, of course, standard schemas to define users. Vendors ship schemas with their LDAP-enabled directory server products that also may include some extensions to support special features they feel are common and useful to their client applications.

Attention: Standard schemas should not be modified. If a standard schema proves to be too limiting for the intended use, it can be extended to support other requirements. Standard schema elements, however, should not be deleted. Doing so can lead to interoperability problems between different directory services and LDAP clients.

Another important issue is to use a consistent schema within the directory server because LDAP-enabled application clients locate entries in the directory by searching for object classes or attributes and their associated values. If the schemas are inconsistent, then it becomes virtually impossible to locate information in the directory tree efficiently.

Namespace

Each entry has a name called a *distinguished name* (DN) that uniquely identifies it. The DN consists of a sequence of parts called relative distinguished names (RDNs), much as a file name consists of a path of directory names in many

operating systems such as UNIX® and Windows®. Entries are organized into a tree-like structure based on their distinguished names. This tree of directory entries is called the Directory Information Tree (DIT). The root DN of a directory tree is called suffix. Entries whose distinguished name contains the DN of another entry as a parent are considered to reside under the latter entry in the hierarchy (that is, the namespace is hierarchical). The namespace definition determines where you should place your objects and attributes. This then determines the DN of your entries. Entries are named according to their position in the DIT. Figure 3-2 shows an example of a DIT.

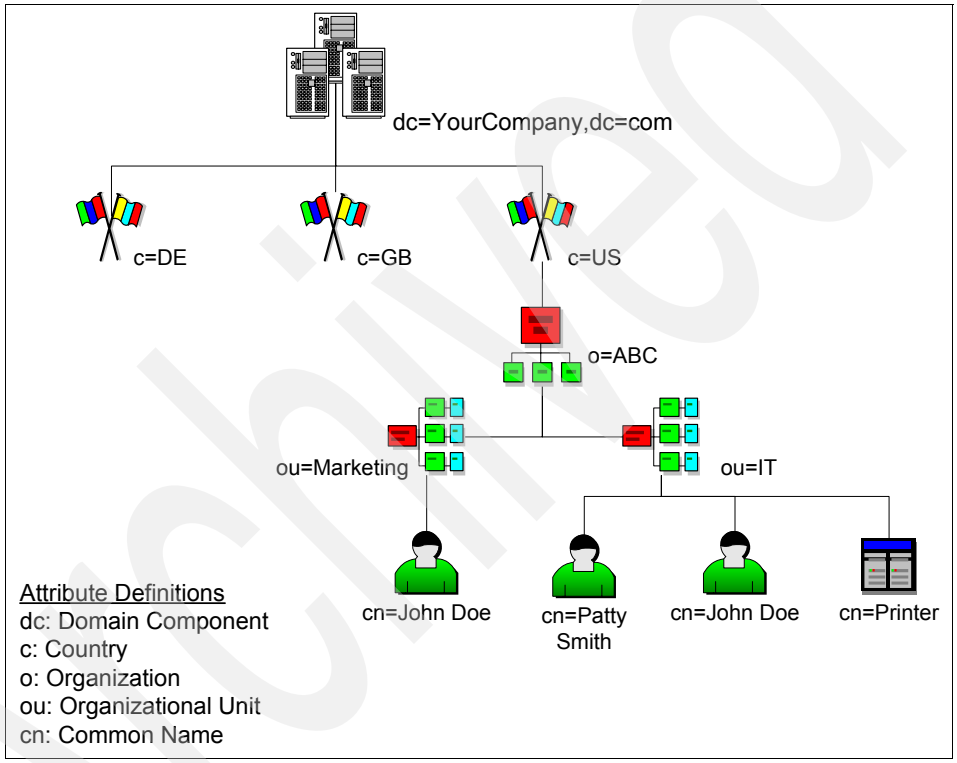


Figure 3-2 Example of a Directory Information Tree (DIT)

DNs are read from leaf to root as opposed to file system names, which usually are read from root to leaf. Each RDN™ is constructed from an attribute (or attributes) of the entry it names.

The DN `cn=John Doe,ou=Marketing,o=ABC,c=US,dc=YourCompany,dc=com` is constructed by adding the RDN `cn=John Doe` to the DN of the ancestor entry `ou=Marketing,o=ABC,c=US,dc=YourCompany,dc=com`. Note that `CN=John Doe` is an attribute in `cn=John Doe Doe,ou=Marketing,o=ABC,c=US,dc=YourCompany,dc=com`. The DN of an entry is specified when it is created. It is legal, though not intuitive,

to create the entry with the DN `mail=jdoe@mail.com,ou=Marketing,o=ABC,c=US,dc=YourCompany,dc=com`. In the example we can see that two `cn=John Doe` entries can exist in the same directory if they have different parent DNs.

The design of the namespace strongly depends on an organization's characteristics and requirements. For example, the choice of creating one or more suffixes can be related to the intent to partition the directory. However, there are some general considerations, as we will discuss.

The flexibility to accommodate changes within the organization is one of the single most important tasks in implementing a directory service. This helps save time and money as the directory service grows. This is the reason why the DIT should be reasonably shallow unless there are strong reasons to design deep branching levels down the directory tree. Even a DIT in which all entries are placed under the same parent DN could be a good solution in many organizations, especially small ones. However, branching could be required for management and performance purposes, so try for a branching methodology that is flexible and that still reflects enough information about the organization.

Because the structure of organizations often changes considerably over time, the aim should be to branch the tree in such a way as to minimize the number of necessary changes to the directory tree once the organization has changed. Note that renaming a top-level entry, for example, has the effect of requiring a change of the DNs of all entries below its branch point. This has an undesirable impact on the service.

Criteria that may be considered when branching the directory tree include:

- Separation of internal (for example, employees) and external (or Internet) users
- Organizational structure, such as departments
- Geographical locations
- Management responsibilities

These first criteria are probably the most intuitive. However these all can lead to a large administrative overhead if the organization is very dynamic and changes often. Other criteria include:

- Performance and system characteristics

  Although it should not be the primary design goal to analyze and meet the strengths and circumvent weaknesses of a specific server (as they may change with new software releases or other vendor products), it is good practice to have some characteristics of the implementation in mind when branching a DIT.

- Human or machine clients

  If users manually type in search criteria, the DIT should provide the information in an intuitive manner.

Remember that the method of storage for the DIT of the directory is implementation-dependent and hidden from the user of that directory. For example, the IBM Tivoli Directory Server uses DB2® as its data store, but no DB2 constructs are externalized to LDAP.

Naming style

The first goal of naming is to provide unique identifiers for entries. Other major goals should be:

- Have user-friendly object and attribute names.
- Let querying of the directory tree be intuitive.
- Allow a user (directory designer, exploiter, or application programmer) to easily understand the conventions used to name the objects.
- Allow a user to adopt the same conventions for naming new schema for his or her own applications.

3.2.9 Physical architecture

Although a directory server can be used for different services, as discussed in 3.2.1, “General definition” on page 53, in this book we refer to it mainly as a *user repository*.

In this section we show where to place directory servers with respect to security domains. Figures in this section show only the directory components. For a discussion of the physical network zones, see 2.3.1, “Localizing a global vision” on page 31, and 2.3.2, “Network zones” on page 34.

We begin with a simple scenario and then we move to more complete topologies.

A directory server can be accessed from many different clients. Many current applications have an embedded LDAP module to support LDAP authentication. For example, common LDAP clients are HTTP servers, application servers, operating systems, Access Manager WebSEAL, and customized applications that use an API.

The directory server, which is the user registry, should be in a restricted access zone, such as a production zone, to which access may be strictly controlled. Firewall configurations should prevent direct access to the user registry from

uncontrolled zones such as the Internet. A simple placement of the directory server is shown in Figure 3-3.

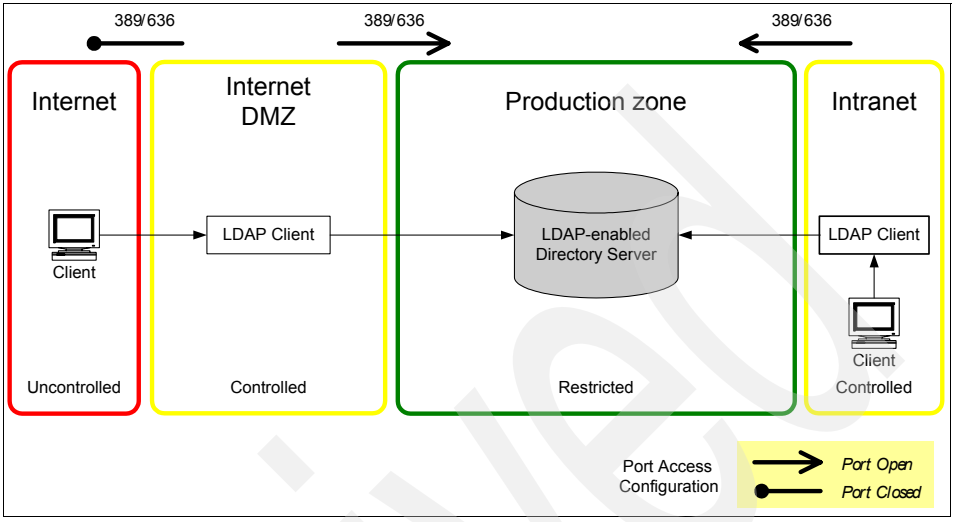


Figure 3-3 Simple architecture with one directory server

Access for ports 389 and 636, or other LDAP ports if not using the LDAP standard ports, should be closed by an Internet-facing firewall, and outgoing LDAP port access should be allowed from the Internet DMZ to another zone only if initiated by specific servers such as WebSEAL, for example.

Now we add one consideration to the simple architecture described above. Because LDAP clients usually require read access to the user registry, it makes sense to use replicas to increase security by separating the *read* functions of the registry from the *write* functions. This can be done by creating a registry replica used for *read-only* access (such as authentication) and leaving the registry master only for making updates. Normal applications need access only a replica server, while the master is accessed solely for administrative tasks. In this configuration the master could be placed in the production zone as well, or it might be placed in the intranet if this is considered secure enough.

In 2.3.2, “Network zones” on page 34, we showed that a *management* secure zone might be introduced to increase security. In this case the read-only replica should be placed in the production zone, while the master should be placed in the management zone. The resulting architecture is shown in Figure 3-4 on page 67.

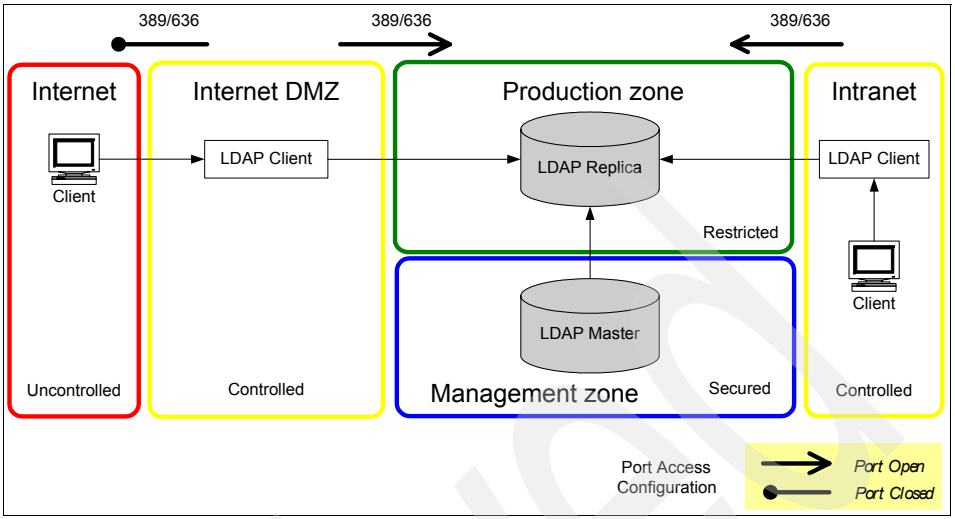


Figure 3-4 Architecture with LDAP master in a management secure domain

Both Figure 3-3 on page 66 and Figure 3-4 use port 389 and 636 together. You could allow only SSL communication, port 636 by default, between Internet DMZ and production zone. You might also consider opening only port 389 between the intranet and the production zone. This has the benefit of relying on firewalls to deny communications on the same port between the Internet DMZ and the intranet. However, the firewall only allows communications from the Internet DMZ to the production zone when the requests come from specified hosts. This approach has the disadvantage of allowing non-encrypted LDAP communications within the intranet. There is no best solution, because it all depends on the actual level of security required in each zone.

Other architectures are also possible according to the namespace structure. In “Namespace” on page 62 we introduced different criteria to choose suffixes and branching. One criteria is to keep Internet and intranet users separated. An Internet or external user can be, for example, anyone who has access to a company’s application available on the Internet. Intranet or internal users are employees and in general people who work for the company. We have already discussed whether it is better to have one user registry or to split the namespace into multiple trees. If an organization decides to have a directory server dedicated to internal users, this could be placed in the intranet. The resulting architecture is shown in Figure 3-5 on page 68. Note that in this topology, the LDAP ports can be closed between the intranet and the production zone. Of course, an additional replica server for the internal users can be used inside the production zone to allow applications to access read-only LDAP information.

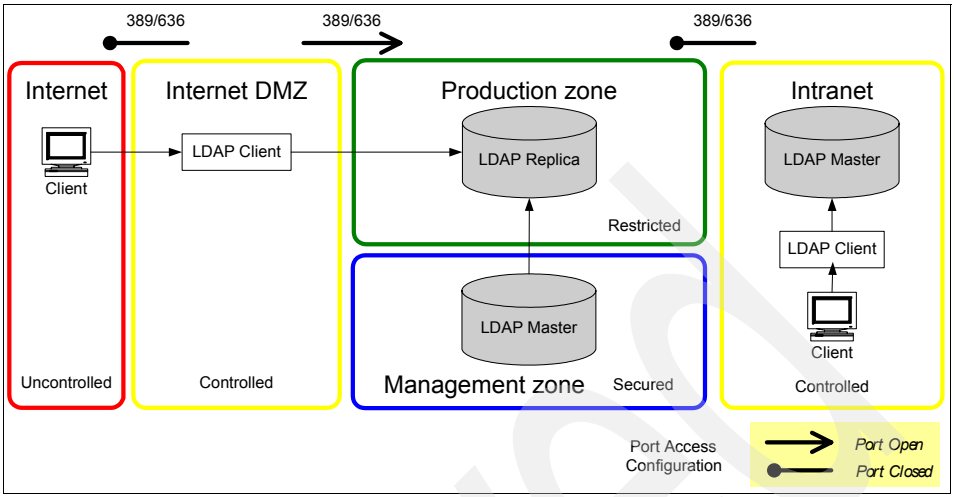


Figure 3-5 Architecture with different internal and external users repository

In addition, an intranet restricted zone separated from the rest of the intranet could be created to have even more security. In this case the LDAP master server for intranet users could be placed there.

An architecture intermediate between those shown in Figure 3-4 on page 67 and Figure 3-5 can be set up by using only one LDAP master in the management zone and replicating different subtrees to different security zones. The subtree with external users is replicated to a server in the production zone, while the subtree with internal users is replicated to a server in the intranet.

We have just introduced the idea of replication by subtree. In the following section we describe the characteristics of replication and partitioning. The use of these methods leads to setting up more complex architectures, but also introduces high availability and scalability.

3.2.10 Availability and scalability

As we introduced in 3.2.6, “Distributed directories” on page 58, the two main methods for improving availability and scalability are replication and partitioning.

Replication

Replication is based on a master-slave replication model. LDAP refers to the master as master server and to the replica as replica server. The database of every replica server contains an exact copy of the master server’s directory data. There is no limit to the number of replicas that can be configured, but replicas can only be read, not updated. A replica server can be promoted as master

server if required (for example, if the master server is out of service for an extended period of time) in order to allow write operations to the directory during this time. Replication has two main benefits:

- Performance: Provides a service from multiple machines in order to satisfy a search as quickly as possible.
- Availability: If one server is temporarily down, the directory service continues to be available from a replicated server.

In distributed environments replicas are also often considered in a geographical perspective. A copy of the data is replicated to each site of a spread company. This local replica protects LDAP services in case of network problems.

Partitioning

Benefits of partitioning a directory tree and distributing it to multiple LDAP servers at multiple locations include:

- Scalability: More data can be accommodated by the directory because the tree information is stored on a collection of servers, not just one. This provides for a (theoretically) indefinite size of namespace.
- Availability: Spreading the directory information into subtrees reduces the possibility of a single point of failure.

  However, a drawback to this approach is that the probability of failure can increase as more systems are involved and depending on how the directory information is accessed. If requests are primarily being handled (and eventually forwarded to other servers) by a single server, the service still depends on a single machine (unless other provisions are in place).
- Performance: The workload of the actual data retrieval can be spread among the servers.
- Manageability: Each location can manage its own part of the directory tree on the local machine. Alternatively, management can also be done centrally.

A technique for partitioning a directory tree is to use LDAP referrals, which point to a different partition of a namespace stored on a different (or the same) server. For example, if your main directory server is located in New York and you want to redirect all requests for `<ou=Austin,o=Your_ORG,c=US>` to a directory server located in Austin, you can specify this with a referral entry in the main directory tree in the following format:

```
ldap://<hostname:port>/ou=Austin,o=Your_ORG,c=US
```

A referral is a pointer to another portion (partition) of a directory. It is returned by the server to a client and it is then up to the client to follow such a referral.

Scalability and performance can be increased easily by combining replica and referral. Also, high availability in read mode can be reached easily using replicas, but replicas do not provide availability in write mode, because only masters can be updated. Two methods for providing availability for the master are:

- Install the master in a clustered environment (for example, using HACMP™ for AIX).
- Use an LDAP server product that allows a topology with more than one master. For an example, see the IBM Tivoli Directory Server peer replication topology in 3.3.5, “Availability and scalability” on page 83.

In Figure 3-6, we show an example of a highly available and scalable LDAP multiple master environment. This figure only considers access from the Internet. For intranet access the consideration made in the previous section can be repeated.

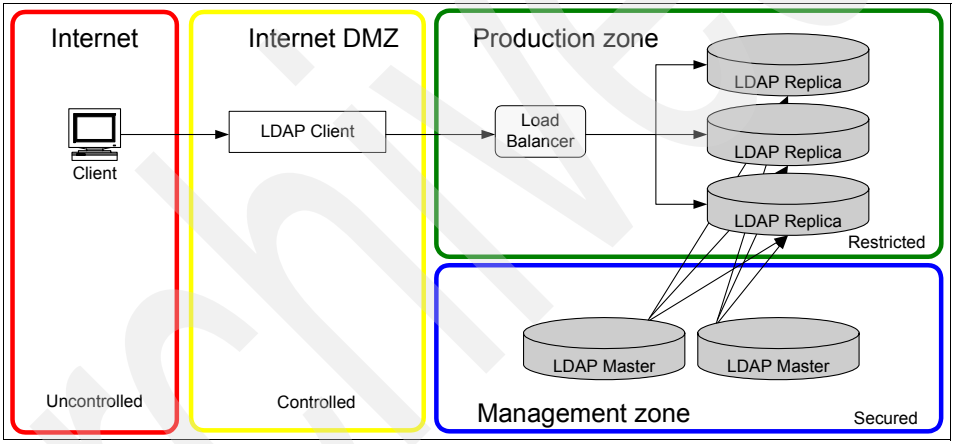


Figure 3-6 Highly available and scalable LDAP multiple master environment

3.2.11 Administration

In this section we show the tools for administering the directory, then we present a brief reflection about who should perform administrative tasks.

The LDAP specifications contained in the pertinent RFCs include functions for directory data management. These include functions to create and modify the directory information tree (DIT) and to add, modify, and delete data stored in the directory.

Vendor products, however, most likely include additional tools for configuring and managing an LDAP server environment. These include such functions as:

- Server setup (initial creation)
- Configuring a directory information tree
- Content management
- Security setup
- Replication and referrals management
- Access control management
- Logging and log file management
- Resource management and performance analysis tools

Depending on specific needs and preferences, LDAP directory administration can be performed several ways. Different vendors offer different administration tools. Although not all vendors provide tools for all methods, in general there are three tools to manage LDAP directories:

- Graphical administration tools
- Command line utilities
- Custom-written applications

Graphical tool features are specific to each vendor, when provided. In 3.3.7, “Administration” on page 90, we describe the IBM Tivoli Directory Server Web Administration Tool.

Command line tools are based on the LDAP Software Development Kit (SDK), which is mainly a set of libraries and header files. Depending on vendors, most SDKs come with a set of simple command line applications, either in source code or as ready-to-use executable programs. These tools were built using the LDAP API functions and thus can serve as sample applications. They enable you to do basic operations, such as searching the directory and adding, modifying, or deleting entries within the LDAP server. Each basic operation is accomplished with a single program such as **`ldapsearch`** or **`ldapmodify`**. By combining these tools using, for example, a scripting language such as Perl, you can easily build up more complex applications. In addition, they are easily deployable in Web-based CGI programs.

As an alternative to using the administration utilities, custom-written administration tools can be used. A developer has several options for accessing LDAP. API library for both for C and Java languages are available. Another approach for custom-written tools is to use the Java Naming and Directory Interface (JNDI) client APIs. Such administration tools might be desirable when typical data administration, such as adding or modifying employee data, is done by non-technical staff. Writing directly to the API layer may also be necessary for applications that need to control the bind/unbind sequence, or, perhaps, want to customize the referral behavior. This is a more difficult approach because the developer must deal with the conversion of the data to the structures that are

sent over the LDAP protocol. Additionally, the developer must be aware of a particular security setup, such as SSL.

Centralized and distributed administration

The directory administrator (the user with the root DN) is, by default, the only person who can administer information in the directory. At times, it will be necessary to allow other users to have administrative privileges on all or portions of the directory. The Directory Information Tree can be divided into administrative areas. Using ACLs, the directory administrator can give other distinguished names full privileges to manage some subsection of the directory. In order to grant a user administrative permission to a subtree, that user DN must be specified in the entry owner attribute of the root of the subtree. The administrative domain will be delimited by the value of an owner inheritance attribute (OwnerPropagate); if it is set to FALSE, the scope of the administrator will be the single entry on which the owner was set, and if OwnerPropagate is set to TRUE, the administrative domain will be the entire subtree unless a new entry owner is specified in a descendant entry. ACLs also allow granting limited administrative privileges to a DN on a subtree or on a specific directory entry. For more details about ACL, refer to 3.2.7, “Directory security” on page 59.

3.3 IBM Tivoli Directory Server

In this section we describe the IBM directory product, IBM Tivoli Directory Server, formerly known as IBM Directory Server or IBM SecureWay® Directory. We refer to the latest version, Version 6.0, but many features are common to the previous versions. New features with respect to the previous versions are clearly stated at the beginning of the *IBM Tivoli Directory Server Release Notes*, SC32-1682.

The IBM Tivoli Directory Server implements the LDAP V3 specifications as defined by the Internet Engineering Task Force (IETF). Therefore, all general LDAP features described in 3.2, “Directories” on page 53 are implemented in IBM Tivoli Directory Server. We do not go into detail about the LDAP protocol, client-server architecture, or network placement, because these topics have already been generally addressed.

IBM Tivoli Directory Server also includes enhancements in functional and performance areas added by IBM. In this section we focus on the main features, especially from the architectural and security points of view. For more details always refer to the IBM Tivoli Directory Server Information Center for all current documentation, which is available online in Portable Document Format (PDF) or Hypertext Markup Language (HTML) format at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.IBMDS.doc/toc.xml`

3.3.1 Overview

The main features of IBM Tivoli Directory Server include:

- Reliable IBM DB2 Universal Database™ V8.1 engine provides scalability to tens of millions of entries, as well as groups of hundreds of thousands of members.
- Broad platform support: Windows, AIX, Linux (IBM eServer™ System x™, zSeries®, pSeries®, and iSeries®), Solaris™, and Hewlett-Packard UNIX (HP-UX) operating system platforms.
- Robust replication capability with many different topologies, which include cascaded replication and peer-to-peer replication with many master servers.
- Ease of management and usability with Web Administration GUI and features such as Dynamic and Nested Groups, along with Sorted and Paged Search Results.
- Tight integration with IBM operating systems, WebSphere® middleware, and Tivoli identity management and security products.

The product can be downloaded from the IBM Tivoli software products Web site:

`http://www.ibm.com/software/tivoli/products/directory-server/`

It is available for all supported platforms and includes all of its base components.

3.3.2 Base components

The base components of IBM Tivoli Directory Server are:

- IBM DB2 as the backing store to provide per-LDAP operation transaction integrity, high-performance operations, and online backup and restore capability. IBM Tivoli Directory Server Version 6.0 currently ships with DB2 UDB v8.1.
- The server executable: idsslapd.
- Tools to administer and configure the directory. These tools rely on the directory administration daemon (idsdiradm), which runs on each server machine and also enables remote management. See 3.3.6, “Logging” on page 90 for a description of the available tools. The main tools are:
  - Web Administration Tool. This is a J2EE™ compliance application installable on IBM WebSphere Application Server and in its Express version, which is provided with IBM Tivoli Directory Server.
  - GUI for configuring the directory and the database: Configuration Tool (idsxcfg).
  - Command line server utilities.

- IBM Tivoli Directory Server Client SDK, which provides the tools required to develop LDAP applications. It includes:
  - Client libraries that provide a set of C-language APIs
  - C header files for building and compiling LDAP applications
  - Documentation that describes the programming interface and the sample programs
  - Sample programs in source form
  - Command line client utilities

Applications utilizing the LDAP protocol access an LDAP-enabled directory according to the client-server architecture we introduced in the previous section. It is important to secure both client-server communication and administrative tasks as it is necessary to guarantee data integrity and confidentiality. The next sections explain how IBM Tivoli Directory Server implements security.

3.3.3 Directory security

In 3.2.7, “Directory security” on page 59, the main concepts about directory security (authentication, integrity, confidentiality, and authorization) were introduced. IBM Tivoli Directory Server supports all three authentication methods described in that section. Here, we focus on secure communications and data encryption, which are key elements for providing secure authentication, data integrity, and confidentiality. In the last part of this section we show how to manage authorization through the use of ACLs.

Authentication

IBM Tivoli Directory Server supports both server and client authentication:

- For server authentication, the IBM Tivoli Directory Server supplies the client with the IBM Tivoli Directory Server’s X.509 certificate during the initial handshake. If the client validates the server’s certificate, then a secure, encrypted communication channel is established between the IBM Tivoli Directory Server and the client application. For server authentication to work, the IBM Tivoli Directory Server must have a private key and an associated server certificate in the server’s key database file.
- Server and client authentication provides for two-way authentication between the LDAP client and the directory server. With client authentication, the LDAP client must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the LDAP client to the IBM Tivoli Directory Server.

Attention: In case you use self-signed certificates for server authentication, you must distribute the server certificate to each client. For server and client authentication using self-signed certificates you also must add the certificate for each client to the server's key database.

When using some Certification Authority to sign those certificates, they only need to be valid and the other part has to trust the CA certificate.

Data encryption can be performed by Secure Sockets Layer (SSL) security, Transaction Layer Security (TLS), or both. In the following paragraph we describe the two mechanisms. They provide secure authentication as well as integrity and confidentiality.

The default TCP/IP ports are those used for LDAP: 636 for SSL encrypted communications and 389 for unencrypted or TLS communications.

Transaction Layer Security

Transaction Layer Security (TLS) is a protocol defined in RFC 2830 that ensures privacy and data integrity in communications between client and server.

TLS is composed of two layers:

- The TLS Record Protocol, which provides connection security with data encryption methods such as the Data Encryption Standard (DES) or RC4 without encryption. The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by the TLS Handshake Protocol. The Record Protocol can also be used without encryption.
- The TLS Handshake Protocol, which enables the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before data is exchanged. TLS is invoked by using the -Y option from the client utilities.

Attention: TLS and SSL are not interoperable. Issuing a start TLS request (the -Y option) over an SSL port causes an operations error.

Secure Sockets Layer

The IBM Tivoli Directory Server has the ability to protect LDAP access by encrypting data with Secure Sockets Layer (SSL) security. When using SSL to secure LDAP communications with the IBM Directory, both server authentication and client authentication are supported.

With server authentication, the IBM Tivoli Directory Server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the IBM Tivoli Directory Server to the LDAP client application.

Certificates

When using server and client authentication in your SSL settings, the server can be configured to check for revoked or expired certificates. When a client sends an authenticated request to a server, the server reads the certificate and sends a query to an LDAP-enabled directory server with a list that contains revoked certificates. If the client certificate is not found in the list, communications between the client and server are allowed over SSL. If the certificate is found, communications are not allowed.

To conduct commercial business on the Internet, you might use a widely known Certification Authority (CA), such as VeriSign, to get a high-assurance server certificate. If you are using the IBM Tivoli Directory Server in an intranet-only environment, you can use a self-signed server certificate without purchasing a VeriSign high-assurance server certificate.

Kerberos

IBM Tivoli Directory Server supports Kerberos Version 1.4 servers, such as the IBM Network Authentication Service for AIX servers and AIX 64-bit clients. More information about the IBM Network Authentication Service is in the IBM Redbook *AIX 5L Version 5.2 Security Supplement*, SG24-6066.

Note: You must have a Kerberos client installed to use Kerberos authentication.

When utilizing the IBM Network Authentication Service, a client (generally either a user or a service) sends a request for a ticket to the Key Distribution Center (KDC). The KDC creates a ticket-granting ticket (TGT) for the client, encrypts it using the client's password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT, using its password. If the decryption is successful, the client retains the decrypted TGT, indicating proof of the client's identity.

The TGT, which expires at a specified time, permits the client to obtain additional tickets that give permission for specific services. The requesting and granting of these additional tickets does not require user intervention.

The Network Authentication Service negotiates the authenticated, and optionally encrypted, communication between two points on the network. It can enable applications to provide a layer of security that is not dependent on which side of a

firewall either client is situated. Because of this, Network Authentication Service can play a vital role in the security of your network.

You need to create an LDAP server servicename in the KDC using the principal name *ldap/<hostname>.<mylocation>.<mycompany>.com*.

The IBM Tivoli Directory Server can be used to contain Kerberos account information to serve as the backing store for a KDC.

SASL mechanisms

Clients can also authenticate using one of the following Simple Authentication and Security Layer (SASL) mechanisms: CRAM-MD5 and DIGEST-MD5. When a client uses Digest-MD5 (see RFC 2831 for details), the password is not transmitted in clear text and the protocol prevents replay attacks.

Integrity

Data integrity is provided by the whole architecture and in particular by IBM DB2 reliability and the LDAP secure communication protocols we described in the previous section.

Confidentiality

Confidentiality is the protection of information disclosure by means of data encryption to those who are not intended to receive it. The most sensitive resource in a user repository is the user password. Besides secure authentication, IBM Tivoli Directory Server provides other functions to increase confidentiality.

Password encryption

IBM Tivoli Directory Server enables you to prevent unauthorized access to user passwords. The administrator may configure the server to encode the *userPassword* attribute values in either a one-way encoding format or a two-way encoding format. Using one-way encryption formats, user passwords may be encrypted and stored in the directory, which prevents clear passwords from being accessed by any users including the system administrators.

The following one-way encryption options are available:

- UNIX crypt
- SHA (Secure Hash Algorithm)

Note: If the UNIX crypt method is used, only the first 8 characters are effective.

For applications that require retrieval of clear passwords, such as middle-tier authentication agents, the directory administrator needs to configure the server

to perform either a two-way encryption or no encryption on user passwords. In this instance, the clear passwords stored in the directory are protected by the directory ACL mechanism.

The Advanced Encryption Standard (AES) is a two-way encryption option. It is provided to allow values of the *userPassword* attribute to be encrypted in the directory and retrieved as part of an entry in the original clear format. It can be configured to use 128-, 192-, and 256-bit key lengths. Some applications such as middle-tier authentication servers require passwords to be retrieved in clear text format, however, corporate security policies might prohibit storing clear passwords in a secondary permanent storage. This option satisfies both requirements.

IBM Tivoli Directory Server provides the following encryption selections:

None No encryption. Passwords are stored in the clear text format.

crypt Passwords are encrypted by the UNIX crypt encrypting algorithm before they are stored in the directory.

SHA-1 Passwords are encrypted by the SHA-1 encrypting algorithm before they are stored in the directory.

AES128 Passwords are encrypted by the AES128 algorithm before they are stored in the directory and are retrieved as part of an entry in the original clear format.

AES198 Passwords are encrypted by the AES198 algorithm before they are stored in the directory and are retrieved as part of an entry in the original clear format.

AES256 Passwords are encrypted by the AES256 algorithm before they are stored in the directory and are retrieved as part of an entry in the original clear format.

In addition to *userPassword*, values of the *secretKey* attribute are always "AES256" encrypted in the directory. Unlike *userPassword*, this encrypting is enforced for values of *secretKey*. No other option is provided. The *secretKey* attribute is an IBM defined schema. Applications may use this attribute to store sensitive data that need to be always encrypted in the directory and to retrieve the data in clear text format using the directory access control.

Password policy

IBM Tivoli Directory Server enables enforcement of a password policy, which is a set of rules that controls the way passwords are used and administrated in the IBM Tivoli Directory Server. These rules are made to ensure that users change their passwords periodically, and that the passwords meet the organization's syntactic password requirements. These rules also can restrict the reuse of old passwords and ensure that users are locked out after a defined number of failed

attempts. All users except the directory administrator and the members of the administrative group are forced to comply with this password policy.

Authorization

ACLs are attributes attached to a directory entry. Administrators use ACLs to restrict or allow access to different parts of the directory, or specific directory entries. When dealing with ACL the terms *object* and *subject* are commonly used. Object is the directory entry that the ACL is applied to, while subject is the directory entry that is given permission or restriction to perform operations on the object. In this section we show more details about how to deal with access control attributes, subjects, objects, and rights. In the last part of the section we describe how access is evaluated.

Access control model attributes

Each object contains its distinguished name as well as a set of attributes and their corresponding values.

The access control model defines two sets of attributes:

- The *entryOwner* information
- The Access Control Information (ACI)

In conformance with the LDAP model, the ACI information and the entryOwner information is represented as attribute-value pairs.

The entryOwner information controls which subjects can define the ACIs. An entry owner also acquires full access rights to the target object. The attributes that define entry ownership are:

- entryOwner - Explicitly defines an entry owner.
- ownerPropagate - Specifies whether the permission set is propagated to the subtree descendant entries.

The entry owners have complete permissions to perform any operation on the object regardless of the aclEntry. Additionally, the entry owners are the only ones who are permitted to administer the aclEntries for that object. EntryOwner is an access control subject. The directory administrator and administration group members are the entryOwners for all objects in the directory by default, and this entryOwnership cannot be removed from any object.

The Access Control Information specifically defines a subject's permission to perform a given operation against certain LDAP objects. The aclPropagate

attributes determine whether an ACI is applied to just a particular entry or to an entry and its subtree. There are two types of ACI:

- Non-filtered ACLs. This type of ACL applies a permission set explicitly to the directory entry that contains them, but may be propagated to none or all of its descendant entries. The default behavior of the non-filtered ACL is to propagate.
- Filtered ACLs. Filter-based ACLs differ in that they employ a filter-based comparison, using a specified object filter, to match target objects with the effective access that applies to them.

Filter-based and non-filter-based attributes are mutually exclusive within a single directory entry.

Subject

A subject is the entity requesting access to operate on an object. It consists of the combination of a DN (distinguished name) type and a DN. The valid DN types are: access ID, group, and role. The DN identifies a particular access ID, role, or group.

Both groups and roles are a collection of names and are similar in implementation, but they are conceptually different. When a user is assigned to a role, there is an implicit expectation that the necessary authority has already been set up to perform the job associated with that role. With group membership, there is no built-in assumption about what permissions are gained (or denied) by being a member of that group.

In addition, pseudo DNs can be specified as subjects. IBM Tivoli Directory Server contains several pseudo DNs, which are used to refer to large numbers of DNs that share a common characteristic, in relation to either the operation being performed or the object on which the operation is being performed.

Three pseudo DNs are supported by LDAP Version 3:

- Access ID: `cn=this`. When specified as part of an ACL, this DN refers to the bindDN, which matches the object on which the operation is performed. For example, if an operation is performed on the object `cn=personA, ou=IBM, c=US`, and the bindDn is `cn=personA, ou=IBM, c=US`, the permissions granted are a combination of those given to `cn=this` and those given to `cn=personA,ou=IBM, c=US`.
- Group: `cn=anybody`. When specified as part of an ACL, this DN refers to all users, even those that are unauthenticated. Users cannot be removed from this group, and this group cannot be removed from the database.

- Group: `cn=Authenticated`. This DN refers to any DN that has been authenticated by the directory. The method of authentication is not considered.

Access targets

Permissions can be applied to the entire object (add child entry, delete entry), to an individual attribute within the entry, or to groups of attributes (Attribute Access Classes) as described in the following.

Attributes requiring similar permissions for access are grouped together in classes. Attributes are mapped to their attribute classes in the directory schema file. These classes are discrete; access to one class does not imply access to another class. Permissions are set with regard to the attribute access class as a whole. The permissions set on a particular attribute class apply to all attributes within that access class unless individual attribute access permissions are specified.

IBM Tivoli Directory Server defines five attribute classes that are used in evaluation of access to user attributes: normal, sensitive, critical, system, and restricted. As examples, the attribute commonName belongs to the normal class, and the attribute userPassword belongs to the critical class. Refer to IBM Tivoli Directory Server product manuals to see how attributes are classified within the five classes.

Access rights

Directory access rights applied to an access target are discrete. One right does not imply another right. The rights may be combined together to provide the desired rights list following a set of rules discussed later. Rights can be of an unspecified value, which indicates that no access rights are granted to the subject on the target object. The rights consist of three parts:

- Action: Defined values are grant or deny. If this field is not present, the default is set to grant.
- Permission: There are six basic operations that may be performed on a directory object. From these operations, the base set of ACI permissions are taken. These are: add an entry, delete an entry, read an attribute value, write an attribute value, search for an attribute, and compare an attribute value. The possible attribute permissions are: read (r), write (w), search (s), and compare (c). Additionally, object permissions apply to the entry as a whole. These permissions are add child entries (a) and delete this entry (d).
- Access target that we described in the previous section.

By default, the directory administrator, administration group members and the master server get full access rights to all objects in the directory except write access to system attributes. Other entryOwners get full access rights to the

objects under their ownership except write access to system attributes. By default all users have read access rights to normal, system, and restricted attributes.

Access evaluation

Access for a particular operation is granted or denied based on the subject's bind DN for that operation on the target object. Processing stops as soon as access can be determined.

The checks for access are done by first checking for entry ownership, and then by evaluating the object's ACI values. If the requesting subject has entryOwnership, access is determined by the above default settings and access processing stops. If the requesting subject is not an entryOwner, then the ACI values for the object entries are checked.

Refer to the *IBM Tivoli Directory Server Administration Guide*, SC32-1339, for more details about the rules used to calculate access rights based on an object's ACLs and requesting DN.

In the following subsection we show an additional feature.

Proxy authorization group

The proxy authorization is a special form of authentication. By using the proxy authorization mechanism, a client application can bind to the directory with its own identity but is allowed to perform operations on behalf of another user to access the target directory. A set of trusted applications or users can access the Directory Server on behalf of multiple users.

The members in the proxy authorization group can assume any authenticated identities except for the administrator or members of the administrative group.

As an example, a client application, client1, can bind to the Directory Server with a high level of access permissions. UserA with limited permissions sends a request to the client application. If the client is a member of the proxy authorization group, instead of passing the request to the Directory Server as client1, it can pass the request as UserA using the more limited level of permissions. What this means is that instead of performing the request as client1, the application server can perform only those actions that UserA is able to access or perform. It performs the request on behalf of or as a proxy for UserA.

Note: The audit log records both the bind DN and the proxy DN for each action performed using proxy authorization.

3.3.4 Schema

A schema is a set of rules that governs the way that data can be stored in the directory. As we described in 3.2.8, “Schema and namespace” on page 62, the schema defines the type of entries allowed, their attribute structure, and the syntax of the attributes.

The IBM Tivoli Directory Server schema is predefined, but it can be modified in case of additional requirements.

IBM Tivoli Directory Server supports standard directory schema as defined in the following:

- The Internet Engineering Task Force (IETF) LDAP Version 3 RFCs, such as RFC 2252 and 2256
- The Directory Enabled Network (DEN)
- The Common Information Model (CIM) from the Desktop Management Task Force (DMTF)
- The Lightweight Internet Person Schema (LIPS) from the Network Application Consortium.

IBM also provides a set of extended common schema definitions that other IBM products share when they exploit the LDAP directory. They include:

- Objects for white-page applications such as ePerson, group, country, organization, organization unit and role, locality, state, and so forth.
- Objects for other subsystems such as accounts, services and access points, authorization, authentication, security policy, and so forth.

3.3.5 Availability and scalability

IBM Tivoli Directory Server enables the use of replica and partitioning to implement high availability and scalability. In this section we focus on the replica mechanism that provides two main benefits: redundancy of information and faster searches (because search requests can be spread among several different servers). However, partitioning is also supported. It can be used as described in 3.2.10, “Availability and scalability” on page 68.

Through replication, a change made to one directory is propagated to one or more additional directories. In effect, a change to one directory shows up on multiple different directories. The IBM Tivoli Directory Server supports an expanded master-replica replication model. IBM Tivoli Directory Server allows

several replication topologies that can fit different requirements. These topologies include:

- Master - replica topology
- Cascading topology
- Peer-to-peer topology
- Gateway topology
- Distributed directory topology using directory proxy server

The expanded model changes the concept of master and replica. These terms no longer apply to servers, but rather to the roles that a server has regarding a particular replicated subtree. A server can act as a master for some subtrees and as a replica for others. The term, *master*, is used for a server that accepts client updates for a replicated subtree. The term, *replica*, is used for a server that only accepts updates from other servers designated as a supplier for the replicated subtree.

In addition, the following features are common to all the topologies:

- Replicating by subtrees. A replica does not have to replicate an entire directory, but can replicate only a part of it. Specific entries in the directory are identified as the roots of replicated subtrees. Each subtree is replicated independently.
- Assignment of server role by subtree. A server can act as a master for some subtrees and as a replica for others.
- Replication scheduling. Updates to other servers can be immediate or scheduled at a desired time.

Master - replica replication

This is the simplest topology, and it enables:

- Increasing performance by spreading requests on multiple servers
- Obtaining high availability in read-only mode
- Scaling the directory server

The terms master and replica apply to the roles that a server has regarding a particular replicated subtree. The term *master* is used for a server that accepts client updates for a replicated subtree. The term, replica, is used for a server that only accepts updates from other servers designated as a supplier for the replicated subtree. It is also possible designate that part of a replicated subtree not be replicated. A master server can have several replicas. Each replica can contain a copy of the master's entire directory, or a subtree of the directory. A simple topology with four replicas is shown in Figure 3-7 on page 85.

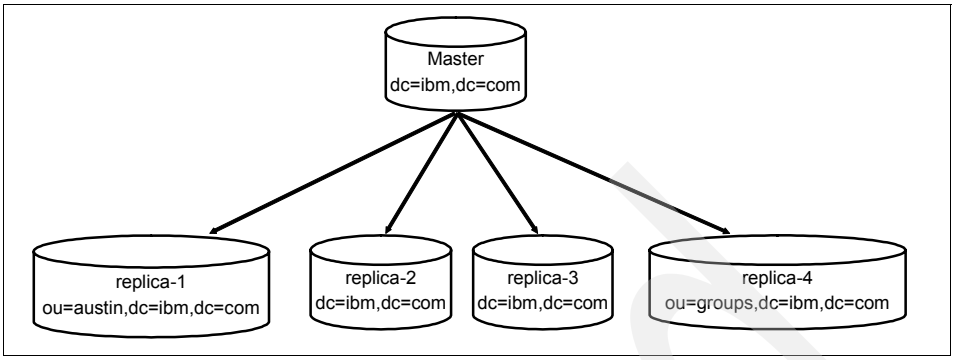


Figure 3-7 Master-replica topology

Updates can be requested on a replica server, but the update is actually forwarded to the master server by returning a referral to the client. If the update is successful, the master server then sends the update to the replicas. Until the master has completed replication of the update, the change is not reflected on the replica server where it was originally requested. If the replication fails, changes are queued up and resubmitted even if the master is restarted. Changes are replicated in the order in which they are made on the master. However administrators can decide to skip specific changes. This can be useful to avoid blocking all replication processes if one update fails, but administrators must remember to fix the problem with the failing update in order to keep the directories synchronized.

The replication process implies that the master binds to the replica using a DN created for that purpose and stored in the Replication Agreement. Three authentication methods are supported:

- Simple bind
- SSL
- Kerberos authentication

Refer to 3.3.3, “Directory security” on page 74 for explanations about these methods.

Cascading replication

This topology adds one element to the previous one, the cascading server. A cascading server, also known as a forwarding server, is a replica server that replicates all changes sent to it. This contrasts to a master server in that a master server only replicates changes that are made by clients connected to that server. A cascading server can relieve the replication workload from the master servers in a network which contains widely dispersed replicas.

The use of cascading servers implies that the master replicates to a small number of forwarders, which in turn replicate to other servers. This enables implementation of cascading replication, as shown in Figure 3-8.

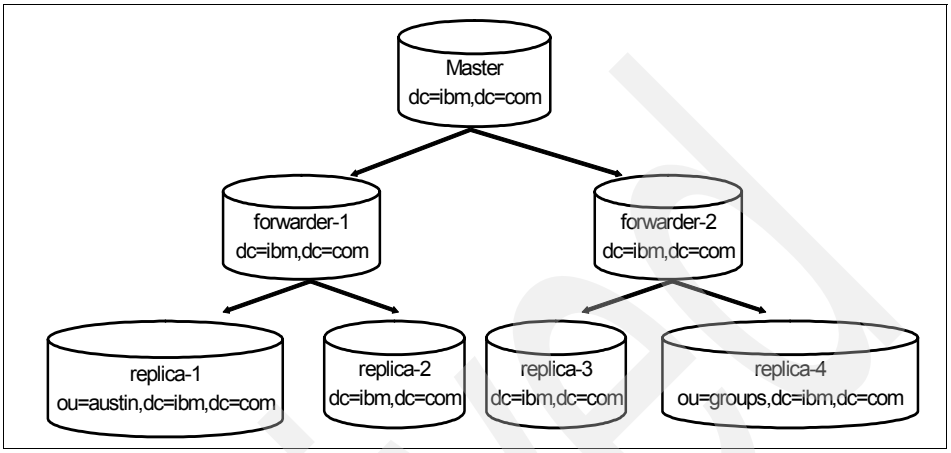


Figure 3-8 Cascading replication

Peer-to-peer replication

In addition to the three benefits pointed out for a master - replica topology, the introduction of peer servers enables setting up multiple master servers. Peer replication can improve performance and master availability. Performance is improved by providing another server to handle updates. For example, this may be useful for setting up a local master in a widely distributed network. Availability is improved by providing a backup master server ready to take over immediately if the primary master fails.

Peer replication topology allows multiple master servers, with each master responsible for updating other master servers and replica servers. A master server is called *peer server* when there are multiple masters for a given subtree. A peer server does not replicate changes sent to it from another master server; it only replicates changes that are originally made on it. A topology with two peer servers and four replicas is shown in Figure 3-9 on page 87.

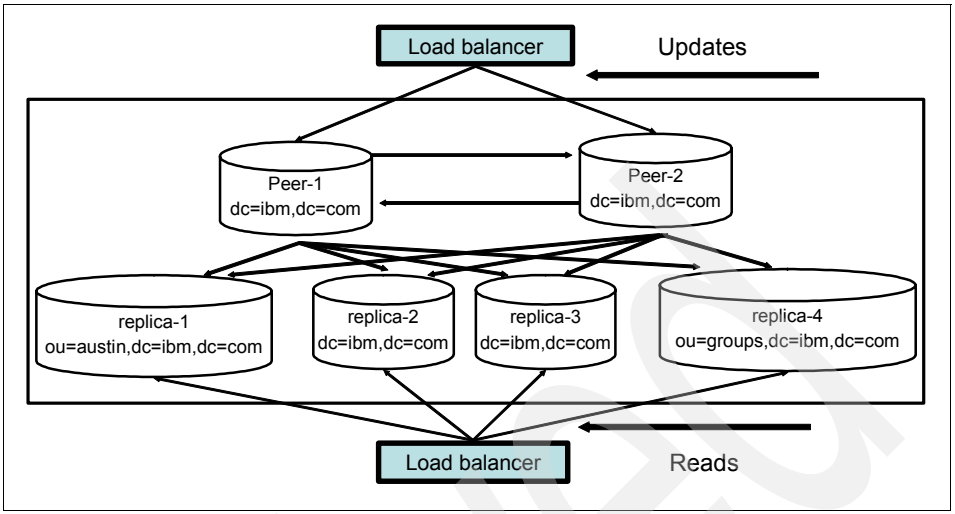


Figure 3-9 Peer replication

Peer and forwarding replication can be combined. For example, two forwarder servers could be added to the topology shown in Figure 3-9.

Conflict resolution is required in peer-to-peer replication to avoid inconsistencies in the directory data. IBM Tivoli Directory Server uses a *time stamp* on add and modify operations to mitigate this. The entry with the most recent modify time stamp on any server in a multi-master replication environment is the one that takes precedence. Replicated delete and rename requests are accepted in the order received without conflict resolution. When a replication conflict is detected the replaced entry is archived for recovery purposes is the Lost and Found log (see 3.3.6, “Logging” on page 90 for more information).

Gateway replication

The introduction of gateway replication enables reduction of network traffic. This is particularly useful in a distributed environment with at least few servers in different locations.

Gateway servers are master servers used to collect and distribute replication information effectively across a replicating network. A gateway server has two functions:

- Collects replication updates from the peer/master servers in the replication site where it resides and sends the updates to all other gateway servers within the replicating network.

- Collects replication updates from other gateway servers in the replication network and sends those updates to the peers/masters and replicas in the replication site where it resides.

Figure 3-10 shows a gateway topology with four sites and four gateway servers.

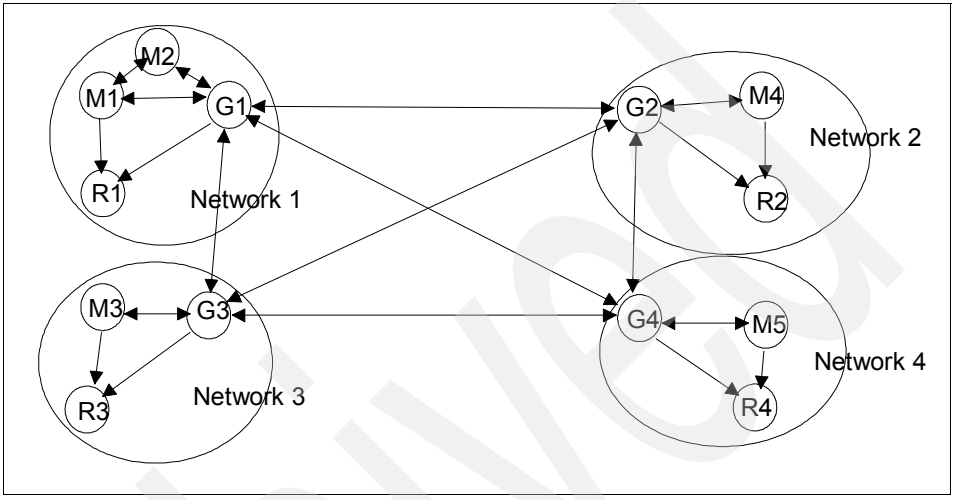


Figure 3-10 Gateway replication

Distributed directory with directory proxy server

IBM Tivoli Directory Server has the ability to be configured either as a standard directory server or as a directory proxy server. A proxy server is a special type of IBM Tivoli Directory Server that provides request routing, load balancing, fail over, distributed authentication, and support for distributed/membership groups and partitioning of containers. Most of these functions are provided in a new backend, the proxy backend. The proxy server does not have an RDBM backend and cannot take part in replication.

A directory proxy server sits at the front-end of a distributed directory and provides efficient routing of user requests thereby improving performance in certain situations, and providing a unified directory view to the client. It can also be used at the front-end of a server cluster for providing fail over and load balancing. The proxy server also provides data support for groups and ACLs that are not affected by partitioning and support for partitioning of flat namespaces.

The proxy server is configured with connection information to connect to each of the backend servers for which it is proxying. The connection information comprises of host address, port number, bind DN, credentials and a connection pool size. Each of the back-end servers is configured with the DN and credentials that the proxy server uses to connect to it. The DN must be a

member of the back-end server's (local) administration group or local administrator. Finally, the proxy server is configured with its own schema. You need to ensure that the proxy server is configured with the same schema as the back-end servers for which it is proxying. The proxy server must also be configured with partition information.

Figure 3-11shows a distributed directory using a directory proxy server.

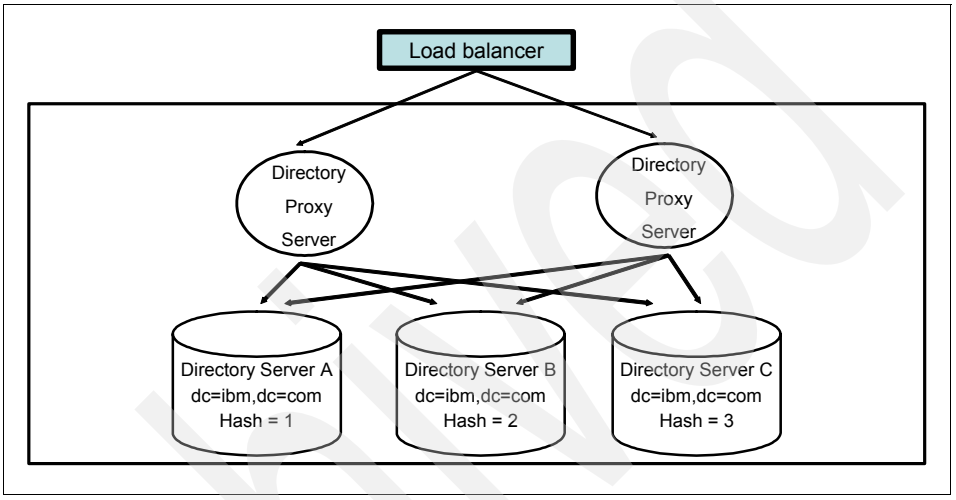


Figure 3-11 Distributed directory with directory proxy server

In this setup, three servers have their data split within a *container* (under some entry in the directory tree). Because the proxy server handles the routing of requests to the appropriate servers, no referrals are used. Client applications need only be aware of the proxy server. The client applications never have to authenticate with servers A, B, or C.

Note: In a distributed directory scenario, each server excluding the directory proxy server may have one or more peers and replicas.

Data is split evenly across the directories by hashing on the RDN just below the base of the split. In this example the data within the subtree is split based on the hash value of the RDN. Hashing is only supported on the RDN at one level in the tree under a container. Nested partitions are allowed. In the case of a compound RDN the entire normalized compound RDN is hashed. The hash algorithm assigns an index value to the DN of each entry. This value is then used to distribute the entries across the available servers evenly.

3.3.6 Logging

The IBM Tivoli Directory Server provides several logging utilities that can be viewed either through the Web Administration Tool or the system command line. Different types of logs are available. They can be configured and activated separately at different levels. Log types include:

- Error log
- Audit log
- Lost and Found log
- DB2 error log
- Bulkload error log
- Administration daemon error log
- Administration daemon audit log

These logs can be used for troubleshooting or for audit purposes.

Additionally, the *Changelog* can be activated. This is a subtree of the directory in which all changes to the directory are recorded. This is very useful for queuing up changes for replication purposes or when the data stored in the directory has to be integrated with other sources. IBM Tivoli Directory Integrator (see 3.5, “IBM Tivoli Directory Integrator” on page 96) uses the Changelog to individuate the data modifications that have to propagate to other repositories.

3.3.7 Administration

Administration tools are one of the outstanding features of IBM Tivoli Directory Server. The main benefits are:

- Remote administration: Administration tools and IBM Tivoli Directory Server can run on different machines.
- Centralized administration: Any directory server can be managed by a single point of control.
- A large number of administrative tasks are available.
- Tools are user friendly and intuitive to use.

Administration tools rely on the directory administration daemon, which must be running continuously on every machine on which IBM Tivoli Directory Server is installed. The directory administration daemon accepts requests by way of LDAP extended operations and supports starting, stopping, restarting, and status monitoring of the IBM Tivoli Directory Server. By default, the IBM Tivoli Directory Server administration daemon listens on two ports (port 3538 for non-SSL connections and port 3539 for SSL connections) if SSL communication is enabled.

Although APIs can be used to develop custom applications to administer directories, the administration tools allow all normally required administrative tasks to perform. As introduced in 3.3.2, “Base components” on page 73, the two main tools are the Web Administration Tool graphical user interface (GUI) and the command line utilities. In addition, the *idsxcfg* utility is useful for performing the initial directory configuration and for managing the DB2 database.

Web Administration Tool

This is a J2EE compliance application installable on an application server, such as the embedded version of IBM WebSphere Application Server - Express included with the IBM Tivoli Directory Server. This application provides a console that can be used to administer all the configured LDAP servers, so only one Web Administration application is required within an organization. As this is a Web application, it can be accessed by a browser without the need any other client.

This application enables administration of several types of LDAP servers. Supported directories are: IBM Tivoli Directory Server 6.0, IBM Tivoli Directory Server 5.2, IBM Directory Server 5.1, IBM Directory Server 4.1, IBM SecureWay Directory 3.2.2, OS/400® V5R3, and z/OS® R4.

Console users

Users accessing the console select a server and provide a user name and a password when logging in. The console administrator, rather than accessing a directory server, can access the console administration interface. From this interface the console administrator can manage only the console, which means that he can perform the following operations:

- Add, modify, or delete a directory server from the list of servers that the console can administrate.
- Define how the console accesses a server by selecting the TCP/IP port and enabling SSL (or not).
- Manage console properties such as security settings.

This console administrator does not have to be defined in any directory. Therefore he has no rights on the servers administered by the console.

All users other than the console administrator can access a directory server by providing a defined user name and password. When logged in, users are authorized to perform tasks according to their permissions as set in the directory ACLs. As shown when we described ACLs in 3.3.3, “Directory security” on page 74, ACLs can be set to allow different administrators to perform tasks with equal or different rights on different or the same directory subtrees. Therefore, there can be several levels of administrators, beginning with the directory administrator, who is owner of every entry. For example, there can be a local

administrator for each subtree if the DIT has been split on a geographical base. A regular user may only have read permission on his data.

To facilitate management of administrators, IBM Tivoli Directory Server allows the use of the *Administration Group*. This is a group of users who are given most of the same directory access as is granted to the IBM Tivoli Directory Server administrator. However, some access to the configuration back end may be restricted from administration group members in order to maintain some security control over administrative users. Members of the Administration Group cannot:

- Modify the Administrator Group itself by adding or deleting members
- Clear or modify the audit logs' settings

Console functions

As stated before, the Web Administration Tool enables an extremely wide range of tasks, such as:

- Basic server administration tasks
- Setting server properties
- Configuring security settings
- Managing the IBM Tivoli Directory Server schema
- Managing replication
- Managing logs
- Managing directory entries
- Managing access control lists
- Managing group, roles, and proxy authorization group
- Performing user-specific tasks

Command line utilities

Server and client command line utilities enable directory server administration without the use of the Web interface. In fact, IBM Tivoli Directory Server provides executables that can perform all of the basic tasks shown in the previous section. For some tasks, administrators can choose to use either the graphical tool or these command line utilities, but some administrators find that more complex tasks can be performed more easily with the Web tool (for example, setting replication topologies and agreements and modifying schemas). Performing certain operations with command line utilities requires more steps and deep knowledge of the directory.

3.4 Virtual directory versus metadirectory

With the growing number of data sources within a corporation's IT environment, new technologies have developed to provide a single, consistent view of identity data. Two of these technologies are the *metadirectory* and the *virtual directory*.

3.4.1 Metadirectory

Due to the complexity of these requirements, custom scripting or application development is not usually affordable or maintainable. It is viable only for solutions that involve only a few point-to-point data flows with minimal requirements for event handling, attribute mappings and minimal logging and error handling capabilities. Meta directories are tools that have emerged to provide a complete set of services tailored to handling these issues. They enable integrators to quickly develop, deploy and maintain, and extend a solution for integrating identity data for infrastructure components and applications.

A metadirectory is not another user directory. It is a toolkit that provides graphical tools systems integrators use to work with information about where data is located, how it can be accessed, how the entries in one store are linked with entries in another, and how the data should flow between different directories and databases. Metadirectory run-time services include connectors (agents) for collecting information from many operating system and application specific sources to integrate the data into a unified namespace.

Meta directories also enforce business rules that specify the authoritative source for attribute values, handle naming and schema discrepancies, and provide data synchronization services between information sources. One of the benefits of a metadirectory is that it can create and maintain a central repository consisting of entries and attributes that are *joined* or aggregated from many other sources. However, a central store for data other than the *metadata* is not required for a metadirectory to provide synchronization services.

Figure 3-12 illustrates the concept of a meta directory.

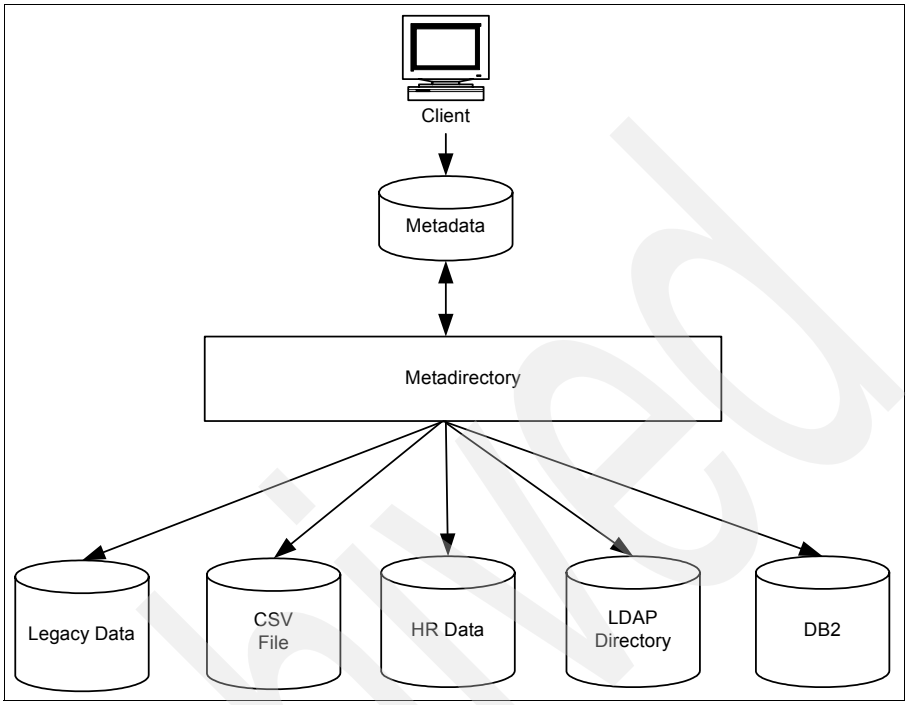


Figure 3-12 Metadirectory

3.4.2 Virtual directories

Virtual directories implement a relatively new, but closely related and complementary to metadirectory, technology with similar services. They provide applications with *virtual views* of the data contained in a variety of data stores. These views can be tailored to the requirements of the application. An application that prefers to use LDAP protocol to access its data can do so, even though the data may be stored in a relational database. Virtual directories are essentially brokers that enable a single query to reference information in multiple data sources dynamically.

A virtual directory could assemble information from multiple sources, perhaps using attributes in a directory as pointers, and then present it to a client application in response to an LDAP query against a virtual directory tree that is defined in the virtual directories *metadata*. A virtual directory provides a layer of abstraction between the applications accessing data and the various repositories where it is stored and managed.

A potential advantage of a virtual directory over a metadirectory, when data access is primarily read-only and there is no need to synchronize data at the various sources, is that data is not moved between sources in order to compose and permanently store an aggregate view. Instead, the data is aggregated as required by the applications that access it. Virtual directories could be appropriate when this is the fundamental requirement, rather than data synchronization, especially for large amounts of data that is mostly read and infrequently written.

In many situations the advantage of a virtual directory can be very difficult to achieve. Directories and databases achieve high performance for portals and security systems that must perform hundreds of authentications and other queries on directory data per second by caching data. Since they control all access to the data, the directory server or database engine can manage a cache efficiently by discarding or replacing cached data when updates are made. Virtual directories can also cache data, of course, but a highly efficient caching strategy is more difficult for them because they do not see updates to the underlying data stores by applications that bypass them and write directly to the data store. When the virtual directory must store cached data persistently due to memory limitations on the server hardware or to provide quick restarts of the server, the distinction between virtual directories and metadirectories is blurred.

Since virtual directories synthesize views of information that can physically reside in several stores with different schemas, they will include most of the functionality of a metadirectory. For this reason, it is likely that metadirectories will evolve to provide some virtual directory services over time. It is likely that over the long term, both metadirectory and virtual directory approaches will have a role in directory integration.

Figure 3-13 illustrates how data is retrieved using a virtual directory.

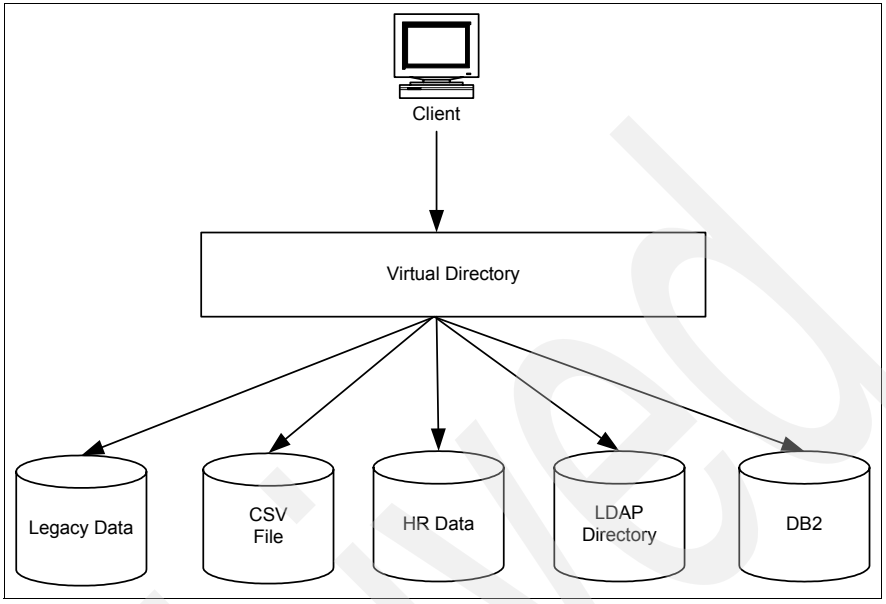


Figure 3-13 Virtual directory

3.5 IBM Tivoli Directory Integrator

There is a need within corporations to create an identity infrastructure consistent across the entire company. It is not important if you only have one repository, many copies of the same repository, or different repositories with redundant data. What is really important is to have consistent and synchronized data throughout the whole organization. Different applications can use data stored in different formats and in different locations, such as LDAP directories, relational databases, flat files, and so on.

The main point is that if one logical object (for example, a user) is defined with some common attributes in more than one place, we want those attributes to have the same values in every place and to be kept synchronized automatically by an integration process flow. The user password is a simple example and is the starting point for implementing a single sign-on solution. The key element for the integration process flows is to clearly define the authoritative data source for each piece of data within the company.

IBM Tivoli Directory Integrator enables you to integrate data from different repositories in an easy and flexible way and can also be used to provide a *metadirectory* or a *virtual directory* services.

In this section we focus on Tivoli Directory Integrator's capability to integrate and synchronize identity data across multiple repositories. Nevertheless, do not be deceived by the word *directory* in its name. IBM Tivoli Directory Integrator enables integration of data from different formats and from different types of repositories, not only from directories. For more detailed technical information, refer to the product manuals, which are available at the following Web site:

http://www.ibm.com/software/tivoli/products/directory-integrator/

In the following sections we first introduce an overview, the main concepts and the main components of IBM Tivoli Directory Integrator, and then focus on security and architecture. Finally we show the logging, monitoring, and administration features.

In this book we refer to IBM Tivoli Directory Integrator version 6.1.1. However, the general concepts and many features are common to the previous releases.

3.5.1 Overview

In 3.1, "Using a centralized user repository" on page 50 we talked about the benefits of a centralized user repository. Nevertheless, we point out that in many circumstances companies prefer (or are obliged) to maintain more than one user repository. This is because it is hard to consolidate all user accounts into only one directory. In fact, the traditional approaches to directory infrastructures might no longer handle the growing volume of users, organizations, and resources in an enterprise. Companies are deploying department-specific applications, each with its own application-specific user repository, resulting in many individual repositories. These repositories can be LDAP directories, relational database (Oracle®, DB2) tables, flat files in different formats (CSV, XML), operating systems, and other.

Companies that decide to maintain more than one user repository and to leverage existing data and tools in order to build a consistent identity and data infrastructure have to integrate them by implementing an identity and data management solution. IBM Tivoli Directory Integrator is designed to fit this requirement.

IBM Tivoli Directory Integrator provides an authoritative, enterprise-spanning identity and data infrastructure critical for security and for provisioning applications, such as portals. It enables integration of a broad set of information into the identity and resource infrastructure. There is virtually no limitation on the type of data or system with which Tivoli Directory Integrator is able to work. It has

a number of built-in connectors to directories, databases, formats, and protocols, as well as an open-architecture Java development environment to extend existing connectors or create new ones, and tools to configure connectors and apply logic to data as it is processed.

In addition to integrating data between applications or directories, IBM Tivoli Directory Integrator can be helpful for other reasons such as:

- Eliminate the need for an inflexible centralized database.
- Capability for distributed data management.
- Supply of a non-intrusive integration. Business and security rules can be introduced to manage flow, ownership, and structure of information between different systems.
- Supply of a modular, flexible, and scalable solution. This is possible because any integration task is divided into simple pieces, which are then linked together. This approach enables introduction of Directory Integrator starting with a portion of the overall solution and then expanding to the whole enterprise. Easy and rapid modifications of the designed solution are always possible.
- Capability of both timed and real-time integration. With the event-driven engine, data flow can be triggered by many types of events such as database or directory change, e-mail arrival, file creation or modification, or HTTP calls.
- Capability to intercept password changes and to propagate the new password to multiple accounts.
- Rapid development, testing, deployment, and maintenance with the graphical interface.
- Support of most standard protocols, transports, APIs and formats such as JDBC™, LDAP, JMS, JNDI, XML, SNMP, and JMX™.
- Support of JavaScript™ for scripting.
- Easy integration with other IBM products such as the WebSphere family and other Tivoli security products such as Access Manager and Identity Manager.
- Wide platform support. Tivoli Directory Integrator can run on UNIX (AIX®, HP-UX, Solaris), Windows, and Linux (RedHat, SUSE, and United Linux on Intel®, IBM System p™, IBM System i™ and IBM System z™). Refer to the *IBM Tivoli Directory Integrator: Administrator Guide,* and the *IBM Tivoli Directory Integrator: Release Notes* for more information about the supported platforms, versions, and requirements.

Figure 3-14 on page 99 shows a general example of an enterprise architecture using IBM Tivoli Directory Integrator. In the following section, we introduce the

Tivoli Directory Integrator's concept and show how information is synchronized and exchanged between the various systems.

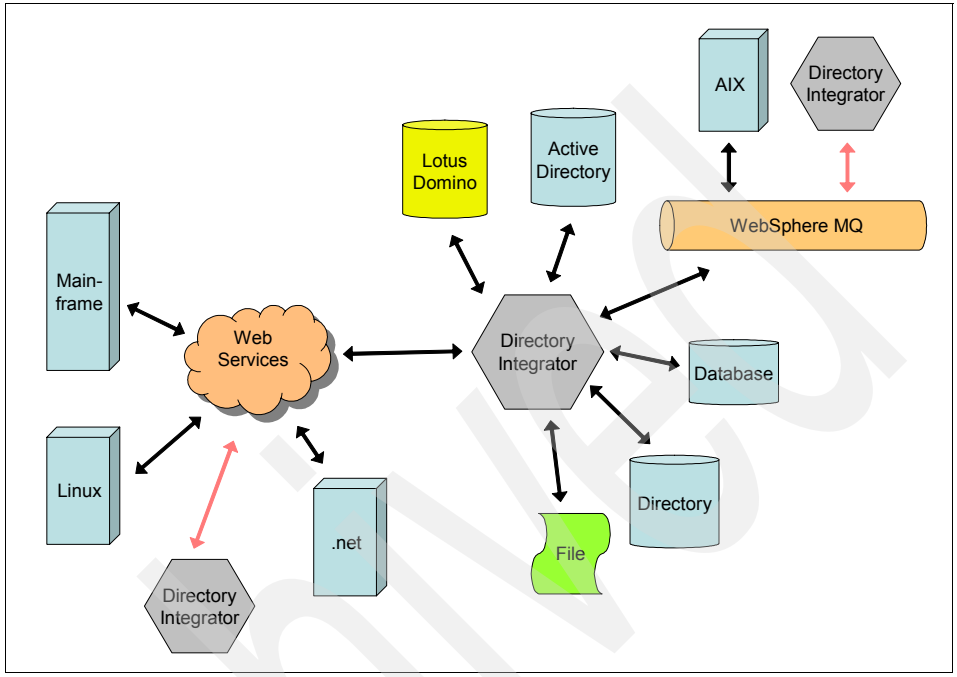


Figure 3-14 A general data integration environment

3.5.2 Concept of integration

The IBM approach is to simplify a large integration project by breaking it into individual small components, then solve it one piece at a time. Integration problems typically can be broken down into three basic parts:

- The systems and devices that have to communicate with each other
- The flows of data among these systems
- The events triggered *when* the data flows occur

These constituent elements of a communications scenario can be described as follows.

Data sources

These are the data repositories, systems, and devices that talk to each other, such as the Human Resources (HR) database, an enterprise directory, the enterprise resource planning (ERP) system, a customer relationship management (CRM) application, the office phone system, a messaging system

with its own address book, or maybe an access database with a list of company equipment and to whom the equipment has been issued.

Data sources represent a wide variety of systems and repositories, such as databases (for example, IBM DB2, Oracle, Microsoft SQL Server), directories (such as Sun Java™ System Directory Server, IBM Tivoli Directory Server, Lotus Domino, Novell eDirectory, and Microsoft ActiveDirectory), files (for example, Extensible Markup Language (XML), LDAP Data Interchange Format (LDIF), or SOAP documents), specially formatted e-mail, or any number of interfacing mechanisms that internal systems and external business partners use to communicate with information assets and services.

Data flows

These are the threads of communications and their content and are usually drawn as arrows that point in the direction of data movement. Each data flow represents a dialogue between two or more systems.

However, for a conversation to be meaningful to all participants, everyone involved must understand what is being communicated. But data sources likely represent their data content in different ways. One system might represent a telephone number as textual information, including the dashes and parentheses used to make the number easier to read. Another system might store it as numerical data.

If these two systems are to communicate about this data, the information must be translated during the conversation. Furthermore, the information in one source might not be complete and might have to be augmented with attributes from other data sources. In addition, only parts of the data in the flow might be relevant to receiving systems.

Therefore, a data flow must also include the mapping, filtering, and transformation of information, shifting its context from input sources to that of the destination systems.

Events

Events can be described as the circumstances that dictate when one set of data sources communicates with another. One example is whenever an employee is added to, updated within, or deleted from the HR system.

An event can also be based on a calendar or a clock-based timer (for example, starting communications every 10 minutes or at 12:00 midnight on Sundays). It can also be a manually initiated one-off event, such as populating a directory or washing the data in a system.

Events are usually tied to a data source and are related to the data flows that are triggered when the specified set of circumstances arises.

In the following section we show how each of these elements is handled by IBM Tivoli Directory Integrator using its base components.

3.5.3 Base components

IBM Tivoli Directory Integrator architecture is based on a set of Java applications, each one with a specific role. Figure 3-15 shows the Tivoli Directory Integrator architecture.

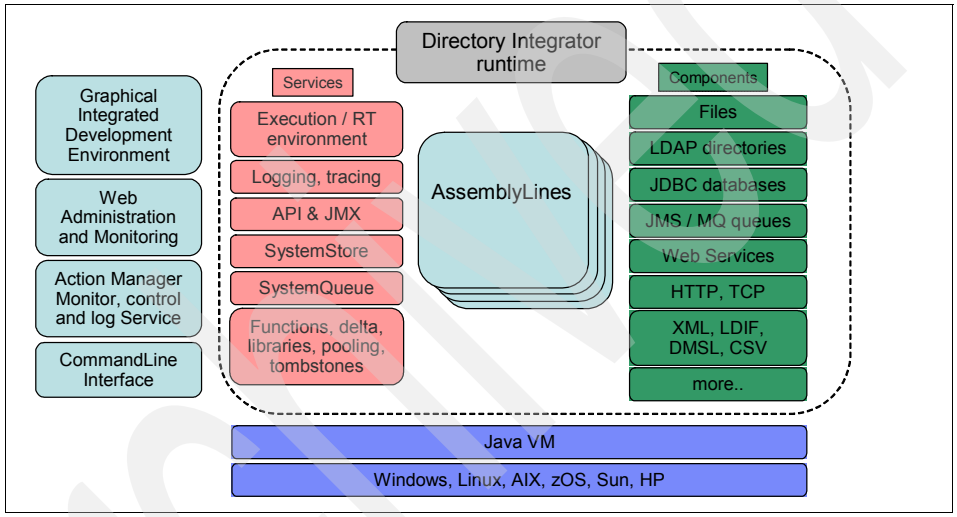


Figure 3-15 Tivoli Directory Integrator architecture overview

Following are the main components:

- Config Editor (CE)

  This program is an Integrated Development Environment (IDE) that provides a graphical interface to create, test, and debug the integration solutions. The Config Editor (CE) creates a configuration file (called a *config*), which is stored as a highly structured XML document and is executed by the run-time engine. The CE is launched by the *ibmditk* script. In 3.5.8, “Administration and monitoring” on page 145 we describe some features of this interface.

- Run-time Server

  Using a configuration file you created with the Config Editor, the Run-time Server powers the integration solution. This application is called *ibmdisrv*, and you can deploy your solution using as many or as few server instances as you want. There are no technical limitations.

From a logical point-of-view the Directory Integrator architecture is divided into the following two parts:

- The *services* system, where most of the system's functionality is provided. Tivoli Directory Integrator services handles log files, error detection, dispatching, and data flow execution parameters. This is also where customized configuration and business logic is maintained. The *Administration and Monitoring Console* (AMC) is the interface for working with these core functionalities. Because it is a Web console, administration can be done remotely using a Web browser, without the need to physically log on to the Directory Integrator server. AMC is described in more detail in 3.5.8, "Administration and monitoring" on page 145.
- The *components*, which serve to provide an abstraction layer for the technical details of the data systems and formats that you want to work with. The two main types of components are *Connectors* and *Parsers*, and because each is wrapped by core functionality that handles things such as integration flow control and customization, the components themselves can remain small and lightweight. For example, if you want to implement your own Parser, you only have to provide two functions: one for interpreting the structure of an incoming bytestream, and one for adding structure to an outgoing one.

This core/component design allows easy extensibility. It also means that you can rapidly build the framework of your solutions by selecting the relevant components and clicking them into place. Components are interchangeable and can be swapped out without affecting the customized logic and configured behavior of your data flows. This means that you can build integration solutions that are quickly augmented and extended while keeping them less vulnerable to changes in the underlying infrastructure.

The key elements of the integration solution are the AssemblyLines. The arrows drawn in Figure 3-14 on page 99 can each represent an AssemblyLine. Each AssemblyLine implements a single unidirectional data flow. A bidirectional synchronization between two or more data sources is implemented by separate AssemblyLines, one for each direction.

AssemblyLines

Real-world industrial assembly line are made up of a number of specialized machines that differ in both function and construction, but have one significant attribute in common: They can be linked to form a continuous path from input sources to output.

An assembly line generally has one or more input units designed to accept whatever raw materials are needed for production (fish fillets, cola syrup, car parts). These ingredients are processed and merged. Sometimes by-products

are extracted from the line along the way. At the end of the production line, the finished goods are delivered to waiting output units.

If a production crew gets the order to produce something else, they break the line down, keeping the machines that are still relevant to the new order. New units are connected in the right places, the line is adjusted, and production starts again. IBM Tivoli Directory Integrator AssemblyLines work similar to real-world industrial assembly lines.

The general philosophy of an AssemblyLine is that it processes data (for example, entries, records, items, objects) from one data source, transforms and combines it with data from others sources, and finally outputs it to one or more targets.

Figure 3-16 shows an example of an AssemblyLine.

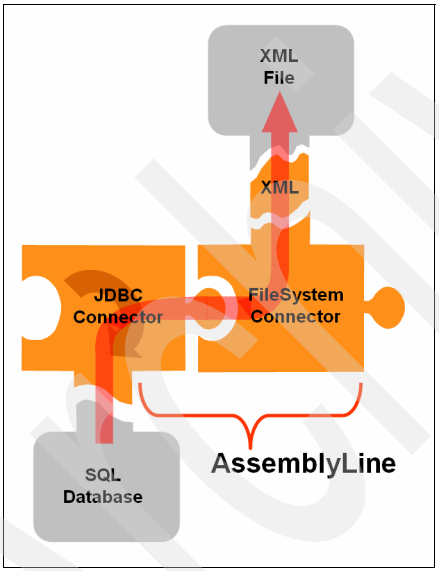


Figure 3-16 AssemblyLine

Let us take a closer look as to what goes on inside an AssemblyLine.

As shown in Figure 3-17 an AssemblyLine may consist of many components. The generic part of the component, called the *AssemblyLine component,* provides kernel functionality like attribute maps, Link criteria, Hooks and so on. The data-source specific part of the component, called the *component interface,* is connected to some system or device, and has the intelligence to work with a particular API or protocol. These component interfaces are interchangeable.

This AssemblyLine *wrapper* makes components work in a similar and predictable fashion. It enables AssemblyLine components to be linked together, as well as providing built-in behaviors and control points for customization.

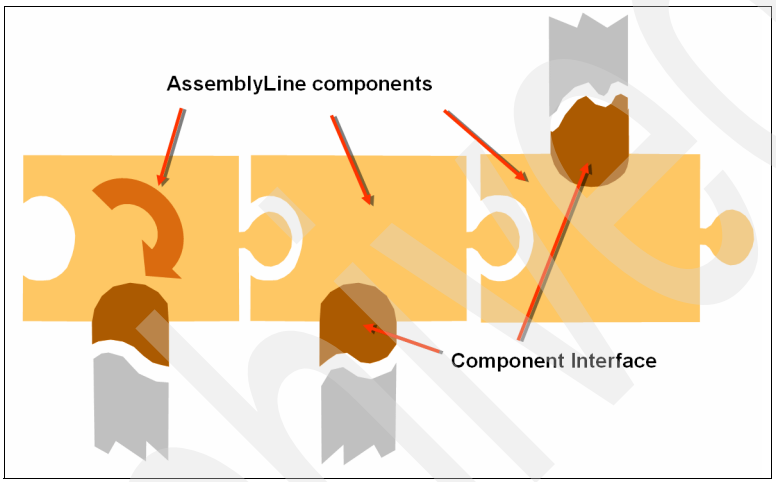


Figure 3-17 AssemblyLine components

How data is organized can differ greatly from system to system. For example, databases typically store information in records with a fixed number of fields. Directories, on the other hand, work with variable objects called *entries*, and other systems use messages or key-value pairs. As shown in Figure 3-18 on page 105 IBM Tivoli Directory Integrator simplifies this issue by collecting and storing all types of information in a powerful and flexible Java data container called a *work entry.* In turn, the data values themselves are kept in objects called *attributes* that the entry holds and manages. The work entry object is passed between AssemblyLine components that in turn perform work on the information it contains, for example, joining in additional data, verifying content, computing new attributes and values, as well as changing existing ones, until the data is ready for delivery to one or more target systems. Additional scripts can also be added to perform these operations.

As a result, attribute mapping, business rules, and transformation logic do not have to deal with type conflicts.

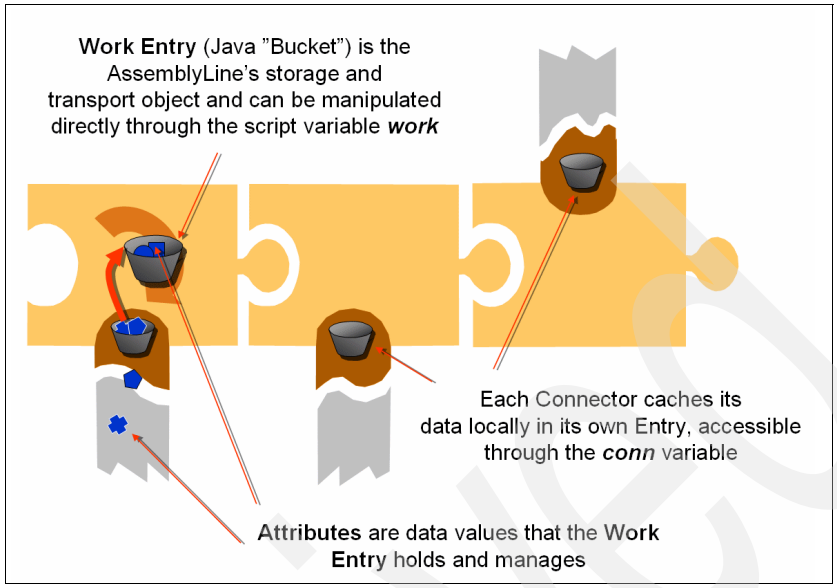


Figure 3-18 Entry objects and Attributes

In addition to the work entry object used by the AssemblyLine to move data down the flow, Figure 3-18 also shows an additional Java bucket nestled in each of the Connectors. These local storage objects are used to cache data during read and write operations. A Connector's local Entry object is called its *conn* object, and exists only within the context of the Connector. When a Connector reads in information, it converts the data to Java objects and stores it in the local *conn* object. During output, the Connector takes the contents of its *conn*, converts this data to native types and sends it to the target system.

However, since each *conn* object is only accessible by its Connector, an additional mechanism is needed to move data from these localized caches to the shared work entry object after Connector input - and the other direction for output Connectors. Figure 3-18 shows an arcing arrow that illustrates this movement of Attributes between the Connectors' local conn Entries and the AssemblyLines work entry object. This process is called Attribute Mapping and is described in more detail in "Attribute Map components" on page 119. Suffice to say that Attribute Maps are your instructions to a Connector on which Attributes are brought into the AssemblyLine during input, or included in output operations.

An AssemblyLine is designed and optimized for working with one item at a time, such as one data record, one directory entry or one registry key. However, if you want to do multiple updates or multiple deletes (for example, processing more than a single item at the time) then you must write AssemblyLine scripts to do this. If necessary, this kind of processing can be implemented using JavaScript,

Java libraries and standard IBM Tivoli Directory Integrator functionality (such as pooling the data to a sorted datastore, for example with the JDBC Connector, and then reading it back and processing it with a second AssemblyLine).

AssemblyLines should contain as few Connectors as possible (for example, one per data source participating in the flow), while at the same time including enough components and script logic to make them as autonomous as possible. The reasoning behind this is to make the AssemblyLine easy to understand and maintain. It also results in simpler, faster, and more scalable solutions.

Connectors

Connectors are like puzzle pieces that click together, while at the same time link to a specific data source.

There are basically two categories of Connectors:

- The first category is where both the transport and the structure of data content is known to the Connector (that is, the schema of the data source can be queried or detected using a well known API such as JDBC or LDAP).
- The second category is where the transport mechanism is known, but not the content structuring. This category requires a Parser (see “Parsers” on page 117) to interpret or generate the content structure in order for the AssemblyLine to function properly.

Each Connector is characterized by two properties, *type* and *mode*. The *type* is related to the data sources that the Connector links to the AssemblyLine. The *mode* identifies the role of the Connector in the data flow, and controls how the automated behavior of the AssemblyLine drives the Component. Connectors can be in one of the following eight modes.

1. Iterator
2. Lookup
3. AddOnly
4. Update
5. Delete
6. CallReply
7. Server
8. Delta

Each Connector mode determines the behavior of a specific Connector, and not all Connectors support all modes of operation. For example, the File System Connector supports only a single output mode, AddOnly, and not Update, Delete or CallReply. When you use a Connector you must first consult the documentation for this component for a list of supported modes. Connectors in Iterator or Server mode are automatically placed in the Feed section of the

AssemblyLine Detail window, Connectors in other modes end up in the Flow section. Each of the connector modes is explained in detail in the next section.

You can change both the type and mode of a Connector whenever you want in order to meet changes in your infrastructure or in the goals of your solution. If you planned for this eventuality, the rest of the AssemblyLine, including data transformations and filtering, will not be affected. That is why it is important to treat each Connector as a black box that either delivers data into the mix or extracts some of it to send to a data source. The more independent each Connector is, the easier your solution will be to augment and maintain.

After a connector is configured for the company environment, it can be transferred to the Connector Library so that any other integration with that specific system or data inherits the configuration of this specific connector in the Connector Library. This saves time and reduces mistakes. Tivoli Directory Integrator also allows usage of external properties to define connector properties and configurations. Connector inheritance and external properties allow ease and consistent changes, reducing system migration impacts and permitting staging of the AssemblyLines in development, QA, and production.

Whenever you need to include new data to the flow, simply add the relevant Connector to the AssemblyLine. In the example of Figure 3-19, there are three connectors: two input connectors to an RDBMS, an LDAP Directory, and one output to an XML document.

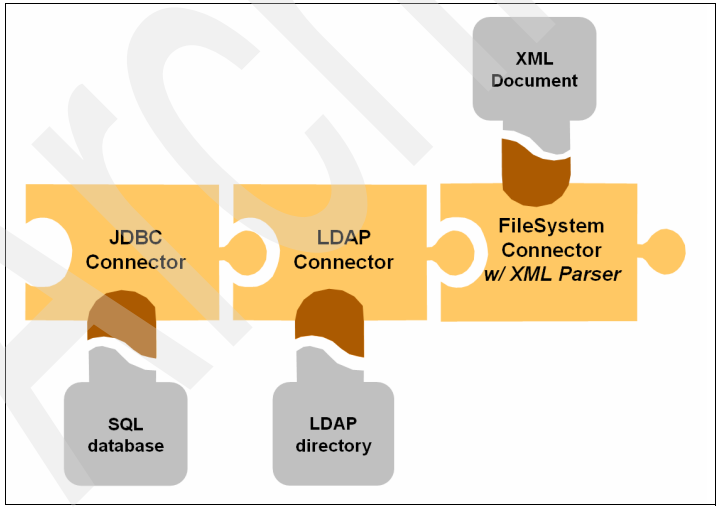


Figure 3-19 AssemblyLine with connectors, parsers, and data sources

Let us examine the different Connector modes.

Connector modes

This section describes in detail each of the eight connector modes.

- Iterator mode

  Connectors in Iterator mode scan a data source and extract its data. The Iterator Connector actually iterates through the data source entries, reads their attribute values, and delivers each Entry to the other AssemblyLine components for processing. A Connector in Iterator mode is referred to as an Iterator.

  Note: It does not matter exactly what the data source is (database, LDAP directory, XML document, and so forth) and how its data is actually stored. Each Connector presents an abstract layer over the particular data source and you access and process data through instances of the Entry and Attribute classes.

  AssemblyLines (except those called with an initial work entry) typically contain at least one Connector in Iterator mode. Iterators (Connectors in Iterator mode) supply the AssemblyLine with data. If an AssemblyLine has no Iterator, it is often useless unless it gets data from another source (for example, the script or process that started the AssemblyLine, or data created in a Prolog script).

  AssemblyLine Connectors that appear in the Feeds section of the component list are driven by the built-in behavior of the AssemblyLine, in order from the top-down. Work Entries fed into the AssemblyLine (from a Feeds Iterator, or passed in from an external system) are passed to the components in the Flow section, executing from the top-down with the Work Entry carrying data down the flow. After the End-of-Cycle is reached, either when the last Flow component has completed or a special call is made like system.skipEntry() or system.exitFlow(), then control is passed back to the top of the AssemblyLine again and the cycle repeats as long as there is more data.

  Multiple Iterators in an AssemblyLine: This has two possible behaviors depending if the Connector in Iterator mode is inside the feed or flow section. If you have more than one Connector in Iterator mode inside the Feed section, these Connectors are stacked in the order in which they appear in the Config (and the Connector List in the Config Editor, in the Feeds section) and are processed one at a time. So, if you are using two Iterators, the first one reads from its data source, passing the resulting work Entry to the first non-Iterator, until it reaches the end of its data set. When the first Iterator has exhausted its input source, the second Iterator starts reading in data.

An initial work entry is treated as coming from an invisible Iterator processed before any other Iterators. This means an Initial work entry is passed to the first Flow section component in the AssemblyLine, skipping all Iterators during the first cycle. This behavior is visible on the AssemblyLine Flow page and Connector mode flowcharts in the product documentation.

Assume you have an AssemblyLine with two Iterators inside the feed section, ItA preceding ItB. The first Iterator, ItA, is used (the AssemblyLine ignoring ItB) until ItA returns no more entries. Then the AssemblyLine switches to ItB (ignoring ItA). If an initial work entry is passed to this AssemblyLine, then both Iterators are ignored for the first cycle, after which the AssemblyLine starts calling ItA.

But if there is a Connector in Iterator mode inside the Flow section, the Iterator will work in the same way as it does in the Feeds, being initialized (including building its result set with the selectEntries call) during AssemblyLine startup, and will retrieve one Entry on each cycle of the AssemblyLine. However, an Iterator in the Flow section will not drive the AssemblyLine flow itself, as it does in the Feeds section.

Sometimes the initial work entry is used to pass configuration parameters into an AssemblyLine, but not data. However, the presence of an initial work entry causes Iterators in the AssemblyLine to be skipped during the first cycle. If you do not want this to happen, you must empty out the work entry object by calling the task.setWork(null) function in a Prolog script. This causes the first Iterator to operate normally.

- Lookup mode

  Lookup mode enables you to join data from different data sources using the relationship between attributes in these systems. A Connector in Lookup mode is often referred to as a Lookup Connector. In order to set up a Lookup Connector you must tell the Connector how you define a match between data already in the AssemblyLine and that found in the connected system. This is called the Connector's Link Criteria, and each Lookup Connector has an associated Link Criteria tab where you define the rules for finding matching entries.

- AddOnly mode

  Connectors in AddOnly mode (AddOnly Connectors) are used for adding new data entries to a data source. This Connector mode requires almost no configuration. Set the connection parameters and then select the attributes to write from the work entry.

- Update mode

  Connectors in Update mode (Update Connectors) are used for adding and modifying data in a data source. For each entry passed from the AssemblyLine, the Update Connector™ tries to locate a matching entry from the data source to modify with the entry's attributes values received.

  As with Lookup Connectors, you must tell the Connector how you define a match between data already in the AssemblyLine and that found in the connected system. This is called the Connector's Link Criteria, and each Update Connector has an associated Link Criteria tab where you define the rules for finding matching entries. If no such entry is found, a new entry is added to the data source. However, if a matching entry is found, it is modified. If more than one entry matches the Link Criteria, the Multiple Entries Found Hook is called so you can script what to do in these cases. Furthermore, the Output Map can be configured to specify which attributes are to be used during an Add or Modify operation.

  When doing a Modify operation, only those attributes that are marked as Modify (Mod) in the Output Map are changed in the data source. If the entry passed from the AssemblyLine does not have a value for one attribute, the Null Behavior for that attribute becomes significant. If it is set to Delete, the attribute does not exist in the modifying entry, thus the attribute cannot be changed in the data source. If it is set to NULL, the attribute exists in the modifying entry, but with a null value, which means that the attribute is deleted in the data source.

  An important feature that Update Connectors offer is the Compute Changes option. When turned on, the Connector first checks the new values against the old ones and updates only if and where needed. Thus you can skip unnecessary updates which can be really valuable if the update operation is a heavy one for the particular data source you are updating.

- Delete mode

  Connectors in Delete mode (Delete Connectors) are used for removing data from a data source. For each entry passed to the Delete Connector, it tries to locate matching data in the connected system. If a single matching entry is found, it is deleted; otherwise, the On No Match Hook is called if none were found or the On Multiple Entries Hook if more than a single match was found. As with Lookup and Update modes, Delete mode requires you to define rules for finding the matching entry for deletion. This is configured in the Connector's Link Criteria tab.

- CallReply mode

  CallReply mode makes requests to data source services (such as Web services) that require you to send input parameters and receive a reply with return values. Unlike the other modes, CallReply gives access to both Input and Output Attribute Maps.

- Server mode

  The Server mode, available in a select number of Connectors, handles events that need to send back a reply message to the system originating the event, providing functionality for building real-time integration solutions.

  These components connect to target systems either polling or subscribing to event notification services.

  On event detection, the Server mode Connector then either proceeds with the Flow section of this AssemblyLine, or if an AssemblyLine Pool has been configured for this AssemblyLine then it contacts the Pool Manager process to request an available AssemblyLine instance to handle this event.

  Once the Server mode Connector has been assigned the AssemblyLine instance it needs to continue, it spawns an instance of itself in Iterator mode, tied to the channel/session/connection that will deliver the event data. This Iterator worker object then operates as any normal Iterator does, including following the standard Iterator Hook flow, reading the event entries one at a time and passing them to the other Flow components for processing until there is no more data to read. At this time, the worker Iterator is cleared away, and if necessary, the Pool Manager is informed that this AssemblyLine instance is now available again.

  When an AssemblyLine with a Server mode connector uses the AssemblyLinePool, the AssemblyLinePool will execute AssemblyLine instances from beginning to end. Before the AssemblyLine instance in the AssemblyLinePool closes the Flow connectors, the AssemblyLinePool retrieves those connectors into a pooled connector set that will be reused in the next AssemblyLine instance created by the AssemblyLinePool (AssemblyLinePool uses tcb.setRuntimeConnector method).

  There are two system properties that govern the behavior of connector pooling:

  – com.ibm.di.server.connectorpooltimeout: This property defines the time-out in seconds before a pooled connector set is released.

  – com.ibm.di.server.connectorpoolexclude: This property defines the connector types that are excluded from pooling. If a connector's class name appears in this comma separated list it is not included in the connector pool set.

When a new AssemblyLine instance is created by the AssemblyLinePool, it will look for an available pooled connector set, which, if present, is provided to the new AssemblyLine Instance as runtime provided connectors. This ensures proper flow of the AssemblyLine in general in terms of hook execution and so on. Note that connectors are never shared. They are only assigned to a single AssemblyLine instance when used.

- Delta mode

  The Delta mode is designed to simplify the application of delta information (make the actual changes) in a number of ways. It provides more optimal handling of delta information generated by either the Iterator Delta Store feature (Delta tab for Iterators), or Change Detection Connectors like the TDS/LDAP/AD/Exchange Changelog Connectors, or the ones for RDBMS and Lotus/Domino changes.

  Note: A Connector in Delta mode needs to be paired with another Connector which provides Delta information, otherwise the Delta mode has no delta information to work with.

  The Delta features in Tivoli Directory Integrator are designed to facilitate synchronization solutions. You can look at the system's Delta capabilities as divided into two sections: *Delta Detection* and *Delta Application*.

  – Delta Detection

    Tivoli Directory Integrator provides a number of change (delta) detection mechanisms and tools:

    - *Delta Store*: This is a feature available to Connectors in Iterator mode. If enabled from the Iterator's Delta tab, the Delta Store feature uses the System Store to take a snapshot of data being iterated. Then on successive runs, each entry iterated is compared with the snapshot database to see what has changed.
    - *Change detection*: These components leverage information in the connected system to detect changes, and are either used in Iterator or Server mode, depending on the Connector. For example, Iterator mode is used for many of the Change Detection Connectors, like those for LDAP, Exchange and ActiveDirectory Changelog, as well as the RDBMS and Domino/Notes Change Connectors. Let us now discuss a few features of change detection connectors.

- *Iterator State Store feature:* This feature uses the System Store to keep track of the starting point for a Change Detection Connector (for example, the changenumber of a directory changelog).

  It keeps track of the next change to be processed, even between runs of the AssemblyLine. The value of the Iterator State Store parameter must be globally unique, so that if you have multiple assembly lines that use Change Detection Connectors, they will each have their own Iterator state data.

- *Change notification feature:* Where supported a Change Detection Connector registers with the data source for change notifications, receiving a signal whenever a change is made. If this parameter is set to *false* the Connector will poll for new changes. If this parameter is set to *true* then after processing all unprocessed changes the Connector will block through the Server Search Notification Control and get notified by the datasource when a change occurs. The Connector will not sleep and time-out when the notification mechanism is used. Other Connectors have to poll the connected system periodically looking for new changes. Those that rely on polling also provide a Sleep interval option to define how often polling occurs.

- *Batch retrieval feature*: Where supported specifies how searches are performed in the changelog. When set to *false* the Connector will perform incremental lookup (backward compatible mode). When set to *true* a query of type *changenumber>=some_value* will be executed for batch retrieval of all modified entries with optional retrieving on pages.

The System Store based Delta Store feature reports specific changes all the way down to the individual values of attributes. This fine degree of change detection is also available when parsing LDIF files. Other components are limited to simply reporting if an entire Entry has been added, modified, or deleted.

This delta information is stored in the work entry object, and depending on the Change Detection component/feature used may be stored as an Entry-Level operation code, at the Attribute-Level or even at the Attribute Value-Level.

– Delta Application (Connector Delta Mode)

The Delta mode is designed to simplify the application of delta information in a number of ways.

Firstly, Delta mode handles all types of deltas, adds, modifies and deletes. This reduces most data sync AssemblyLines to two Connectors, One Delta Detection Connector in the Feeds section to pick up the changes, and a second one in Delta mode to apply these changes to a target system.

Furthermore, Delta mode will apply the delta information at the lowest level supported by the target system itself. This is done by first checking the Connector Interface to see what level of incremental modification is supported by the data source. If you are working with an LDAP directory, then Delta mode will perform Attribute value adds and deletes. In the context of a traditional RDBMS (JDBC), then doing a delete and then an add of a column value does not make sense, so this is handled as a value replacement for that Attribute.

Note: The only Connector that supports incremental modification is the LDAP Connector, since LDAP directories provide this functionality.

This is dealt with automatically by the Delta mode for those data sources that support this functionality. If the data source offers optimized calls to handle incremental modifications, and these are supported by the Connector Interface, then Delta mode will use these. On the other hand, if the connected system does not offer intelligent delta update mechanisms, Delta mode will simulate these as much as possible, performing pre-update lookups (like Update mode), change computations and subsequent application of the detected changes.

Connector states

The state of a Connector determines its level of participation in the operation of the AssemblyLine. In general terms, an AssemblyLine performs two levels of Connector operation:

- Powering up the Connector at the start of AssemblyLine operations and closing its connection when the AssemblyLine completes.
- Driving the Connector during AssemblyLine operation according to the Connector mode.

There are three resulting connector states from these operations:

- Enabled state

  Enabled is the normal Connector state. In the Enabled state, a Connector is powered up and closed, as well as being processed during AssemblyLine operation.

- Passive state

  Passive Connectors (Connectors in Passive state) are powered up and closed just like Enabled Connectors. However, they are not driven by the AssemblyLine automated behavior. However, Connectors in passive state can be invoked by script code from any of the control points for scripting provided by Directory Integrator. For example, if you have a Passive Connector in your AssemblyLine called myErrorConnector then you could invoke it's add() operation with the following script code:

```
var err = system.newEntry(); // Create new Entry object
err.merge(work); // Merge in attributes in the work Entry
// This next line sets an attribute called Error
err.setAttribute ( "Error", "Operation failed" );
myErrorConnector.add( err ) // Add new err Entry;
```

- Disabled state

  In Disabled state, the Connector is not initialized (and closed) or operated during normal AssemblyLine activation. If you want to use it in your scripts, then you must initialize it yourself.

  The name of a disabled Connector is registered but pointing at null, so you can write conditional code like the following example to handle the situation where you plan on setting myConnector to disabled state.

```
if (myConnector != null)
   myConnector.connector.aMethod();
```

  This state is often used during troubleshooting in order to simplify the solution while debugging, helping to localize any problems.

  Directory Integrator provides a library of Connectors to choose from, such as LDAP, JDBC, Microsoft Windows NT4 Domain, Lotus Notes®, and POP3/IMAP. If you cannot find the one you need, you can extend an existing Connector by overriding any or all of its functions using JavaScript. You can also create your own, either with a scripting language inside the Script Connector wrapper or originate with Java.

  Furthermore, Directory Integrator supports most transport protocols and mechanisms, such as TCP/IP, FTP, HTTP, and Java Message Service (JMS)/message queuing (MQ). It also supports secure connections and encryption mechanisms as shown in 3.5.4, "Security capability" on page 125.

  Table 3-2 on page 116 summarizes the more relevant built-in connectors. However, this list can change with the product version. For more information about available connectors, scripting languages, and how to create your own, see the *IBM Tivoli Directory Integrator: Reference Guide*.

Table 3-2 Main available connectors

Applications	PeopleSoft®, SAP®, Siebel® ERP, IBM Tivoli Access Manager.
Databases (using ODBC, JDBC)	Oracle, Microsoft Access and SQL Server, IBM DB2, IBM Informix® and any other database with a valid JDBC driver.
Directories (using LDAP)	CA eTrust, Critical Path, IBM Tivoli Directory Server, iPlanet, Microsoft Active Directory and Exchange, Nexor, Novell eDirectory, OpenLDAP, Oracle, Siemens and any other directory server supporting the ldapV3 protocol.
Directories (using DSMLv2)	IBM Tivoli Directory Server, Novell eDirectory and any other directory server supporting the DSMLv2 protocol.
Files, Streams and Internet Protocols	CSV, XML, DSML, HTTP, LDIF, SOAP, DNS, POP, IMAP, SMTP, SNMP.
Specific Technologies and APIs	Microsoft ADSI, CDO, and other COM; Microsoft NT domains; Lotus Domino directory and databases; Java APIs; system commands.
Messaging Services	IBM MQ, Sonic MQ, and other JMS-compliant systems.
Web Services	Direct with SOAP over HTTP. Note that SOAP over other protocols can be easily addressed using the SOAP Parser or SOAP Function components.
Command Line	Execute commands locally to the execution runtime environment.
Remote Command Line	Execute commands remotely through SSH or RSH protocols.
Changes & Deltas	LDAP Changelog, Active Directory changes, NT/AD Password sync, TCP connections, HTTP gets and posts.
AssemblyLine connector	Runs another AssemblyLine as a connector. It's operation mode depends on the AssemblyLine being called.

Connector Pooling

AssemblyLines can have multiple connectors to the same data source, which can lead to performance problems especially when multiple connectors are being initialized and even more when an AssemblyLine is started on a scheduled basis.

The Connector Pooling feature of Tivoli Directory Integrator creates a number of instances of a single connector, so there is no performance hit when AssemblyLines are initiated or started when they are configured to use connectors residing in pools.

Connector Pools are defined with a minimum and maximum size and they grow on-demand until the maximum size. If the connector is not used anymore, the pool shrinks back to the minimum size configured based, in an also configured amount of time.

If connectors loose their connection with the back end, they are reconnected through the reconnect feature, as discussed in “Automatic connection reconnect” on page 134.

Attention: When the connect or pool maximum size is reached, a new AssemblyLines will fail to start the connector.

Parsers

Even unstructured data, such as text files and bytestreams coming over an IP port, is handled quickly and simply by passing the bytestream through one or more Parsers. The system is shipped with a variety of Parsers, including LDIF, Directory Services Markup Language (DSML), XML, comma-separated values (CSV), SOAP, and fixed-length field. As with Connectors, you can extend and modify these, as well as create your own.

In the example in Figure 3-19 on page 107, a Parser is used to interpret and translate information from an LDIF file. The extracted information is converted into a Java object with a canonical data format so that the LDIF Connector can work with this object and dispatch it along the AssemblyLine.

Now that we introduced the main components of an AssemblyLine, we can show how to customize the AssemblyLine in order to add business rules and logic.

Hooks

Hooks enable developers to describe certain actions to be executed under specific circumstances or at any desired points in the execution of an AssemblyLine. For example, Hooks can be placed before or after a Connector, or in consequence of a specific event such as an update failure or a read success. Directory Integrator automatically calls these user-defined functions as the AssemblyLine runs.

The majority of the scripting in Directory Integrator takes place in the Hooks. For example, Hooks can be used to build custom logic, to handle Global Variables, and to set specific error processes and logs in Hooks.

Scripts

A key capability of IBM Tivoli Directory Integrator is the ability to extend virtually all of its integration components, functions, and attributes through scripts or Java. Scripting can occur anywhere in the system to add or modify the

components of an AssemblyLine. Connectors, Parsers, Functions, and Hooks can be customized in order to perform requested tasks. Scripts are commonly used to map attributes, transform data, access libraries (for example to call Java classes), handle errors, control data flow, and in general to add business logic.

Directory Integrator supports JavaScript plug-in scripting language and extensive script libraries.

Function components

An Function component is an AssemblyLine wrapper around some function or discreet operation, allowing it to be dropped into an AssemblyLine as well as instantiated/invoked from script. The idea behind Function components is to allow complex components (for example, the Web Services connector) to be split into smaller logical units and then strung together as needed, as well as to provide more visual *helper* objects where custom scripting was necessary before. Function components also offer other functionality like launching AssemblyLines, invoking Parsers, and so on. As with all Directory Integrator components, the user can easily create their own Scripted Function components, turning custom logic into a library of reusable AssemblyLine components.

Function components are similar to Connectors in CallReply mode in that they have both Input and Output maps. The Output Map is used to pass parameters to the Function component, while the Input Map lets you retrieve and manipulate return data.

```
myFunction.callreply( work )
```

The above example is invoking the AssemblyLine Function called *myFunction*. Note that calling the AssemblyLine Function method callreply() will cause Attribute Maps and the normal Function Component Hook flow to be executed.

Like the other components, Function Components have a library folder in the Config Browser where you can configure and manage your Function Component library. These can be then dragged into AssemblyLines or chosen from the selection drop-down that appears when you press the Add Component button under the AssemblyLine Connector List.

Also like the other components, Function Components have an Interface part (like the Connector Interface or Parser Interface, in the case of Function Components called the Function Interface) that implements the function logic. When an Function Component is dropped into an AssemblyLine, it is wrapped in an AssemblyLine Function object which provides the generic functionality necessary for the AssemblyLine to manage and execute it.

Also like Connectors, Function components have a State which can be set to Active, Passive or Disabled. State behavior is identical with that of Connectors.

Since Function components are registered as script variables (beans) when the AssemblyLine starts up, you can access them directly from your script using the name given them in the AssemblyLine.

Attribute Map components

This component lets you define Attribute transformations as freestanding Attribute maps that can be stored in your component Library and dropped into your AssemblyLine.

Adding new Attributes to the work Entry and other data manipulation can be quickly performed using the Attribute Map component, which defines a mapping from the work entry to itself, allowing you to create new Attributes as well as change existing ones. And all Attributes defined in Attribute Map components are displayed in the work entry list as well, easing maintenance and support for the configuration.

Controlling AssemblyLine Flow

Tivoli Directory Integrator provides flow components that allows you to define alternate routes in an AssemblyLine. These components act as programming statements, which means AssemblyLines do not need to be simple, unidirectional flows.

Branch components

Branches allow the user to define alternate routes in an AssemblyLine like IF, ELSE, and ELSE IF statements. The Branch provides an interface that allows you to define Simple Conditions based on Attributes in the work Entry object. Multiple Conditions are ANDed or ORed, depending on the Match Any check box setting.

After Simple Conditions are processed, there is a script editor window at the bottom of the Branch details page where you can create your own Condition in JavaScript. The condition is to write in JavaScript language and you must populate ret.value with either a true or false value in order to control the outcome of Condition evaluation. Scripted Conditions can be combined with Simple ones, or used exclusively.

If a Condition evaluates to true then all components attached to the Branch are executed.

After Branch component execution is complete, control is passed to the first component appearing in the AssemblyLine Component List after the Branch.

In Figure 3-20 an AssemblyLine with two branches is shown. In this example, the AssemblyLine will do the following:

- Read a set of users from an XML file. For each user it will execute the following flow:
  - Lookup the LDAP Directory to check if the user exists. If user exists it will map its ldap entry distinguished name in the work object entry with the value ‘true‘.
  - If the user exists (IF branch)
    - Update the user entry
  - If the user does not exist (ELSE branch)
    - Add the user entry to the LDAP Directory
    - Create the user Badge to enter the company

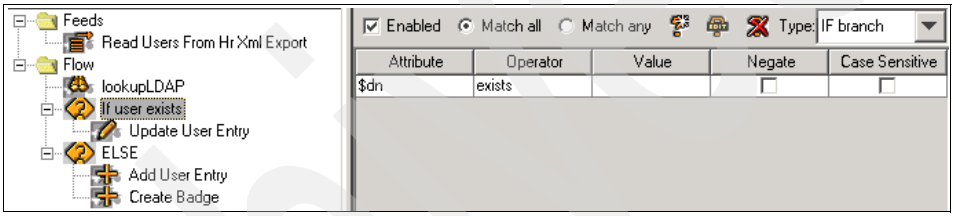


Figure 3-20 AssemblyLine flow using IF and ELSE branches

Switch and Case components

AssemblyLine Switch and Case components allow you to implement switch and case statements as with computer languages that support it.

The Switch component is always on top of Case components and can do switch statements for a Work Attribute, AssemblyLine Operations, Work entry operations for the delta entry and any user defined specific value/expression.

The Case component always follows the Switch component and has any set of components below it.

In Figure 3-21 on page 121 an AssemblyLine with a Switch and the necessary Case components is shown. In this example, the AssemblyLine will do the following:

- Read a set of users from an XML file. In this case it has the Delta enabled as explained in “Delta mode” on page 112.

- Switch the Work entry operation for each changed user received from the iterator:
  - Case Work entry operation is ADD
    - Add the user entry to the LDAP Directory
    - Create the user Badge to enter the company
  - Case Work entry operation is MODIFY
    - Update the user entry

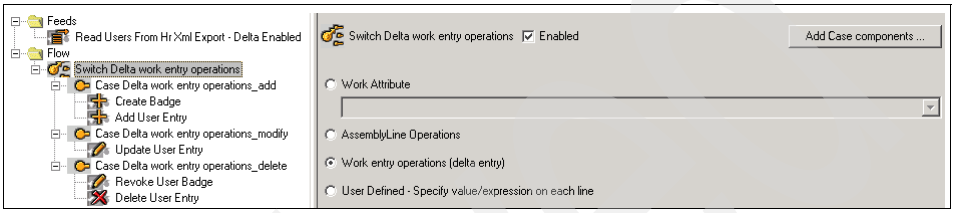


Figure 3-21 AssemblyLine flow using Switch and Case components

AssemblyLine Operations

AssemblyLine Operations allow you to implement any number of distinct functions to be performed by an AssemblyLine. Each Operation has an associated set of Input and Output Maps for defining both parameter values passed in when an Operation is called, as well as Attributes returned after the called AssemblyLine Operation is finished.

After you define Operations for an AssemblyLine, the Switch-Case constructs let you easily implement the logic in the AssemblyLine to deal with them. Furthermore, both the AssemblyLine Function (FC) and the AssemblyLine Connector support AssemblyLine Operation calls. AssemblyLines with Operations can be published as “Adapters”, using the AssemblyLine Publishing feature. These Adapters show up as Connectors and can easily be added to other AssemblyLines or Config Connector Libraries.

For example, if you drop the Web Service Receiver Server Connector into an AssemblyLine, it can generate the WSDL for the AssemblyLine based on its Operations and the associated Attributes.

AssemblyLine Operations are also accessible through API calls. As a result, the Command Line Interface for Tivoli Directory Integrator and the Administration and Monitoring Console both offer features for calling specific AssemblyLine operations and for passing Attribute values between the calling and the called AssemblyLine.

Loop components

The Loop component provides functionality for adding cyclic logic within an AssemblyLine. Loops can be configured for three modes of operation:

- Conditional

  Here you can define Simple and/or Scripted Conditions that control looping. The details window for this type of Loop construct is the same as for the Branch component described in the previous section.

- Connector

  This method lets you set up a Connector for Iterator or Lookup mode, and will cycle through your Loop flow for each Entry returned. This is the preferred way of dealing with Multiple Entries found for a Lookup. The Details pane of this type of Loop will contain the Connector tabs necessary to configure it, connect and discover attributes and set up the Input Map.

  Note that you have a parameter called Init Options where you can instruct the AssemblyLine to either

  - Do Nothing, which means that the Connector will not be prepared in any way between AL cycles.
  - Initialize and Select/Lookup, causing the Connector to be re-initialized for each AL cycle.
  - Select/Lookup Only, to keep the Connector initialized, but redo either the Iterator select or the Lookup, depending on the Mode setting.

  Note also there is a Connector Parameters tab which functions similar to an Output Map in that you can select which Connector parameters are to be set from work Attribute values.

- Attribute Value

  By selecting any Attribute available in the work entry, the Loop flow will be executed for each of its values. Each value is passed into the Loop in a new work entry attribute named in the second parameter. This option allows you to easily work with multi-valued attributes, like group membership lists or e-mail.

System Store

The *System Store* addresses the various needs of Tivoli Directory Integrator for persistent storage. It uses, by default, the DB2Java (CloudScape) RDBMS as its underlying storage technology. Other relational databases, like IBM DB2, can be used to hold the System Store.

The System Store can be shared by multiple instances of Tivoli Directory Integrator servers if the CloudScape database runs in networked mode, or if a multi-user relational database system is used. If CloudScape runs embedded in

an IBM Tivoli Directory Integrator server, it cannot be shared simultaneously with other servers.

The system store implements three types of persistent stores for IBM Tivoli Directory Integrator components:

- The User Property Store
- The Delta Tables
- The Checkpoint/Restart Tables (deprecated)

Each store offers its own set of features and built-in behavior, as well as a callable interface that users can access from their scripts, for example, to persist their own data and state information.

User Property Store

The User Property Store is a System Store table used for maintaining serialized Java objects associated with a key value. This is where persistent component parameters and properties (such as the Iterator State Store) are maintained, as well as data you store.

For example, when you set the Iterator State Store parameter for the Active Directory Changelog Connector, you are specifying the key value that the Connector uses to save and restore Iterator state. If you want the Iteration to start with the first (or last) change entry, simply delete the Iterator State Store entry in the User Property Store.

You can persist your own objects in the User Property Store; however, you can also create and use your own stores using the Store Factory.

Any object to be persisted in the User Property store must be serialized.

Delta Store

The Delta Store is found under the Delta Tables folder in the System Store Browser. Each table represents one Delta Store parameter setting (in the Delta tab of an Iterator). There are a number of classes and methods for working directly with the Delta Store, although this is not recommended.

Tip: It is common to have dozens of AssemblyLines in a company. A central system store with high availability can ease your Tivoli Directory Integrator management.

Password synchronization

The password synchronization feature, which is more a module than a component, can be very useful when designing an AssemblyLine that has the goal to synchronize passwords.

Password synchronization can be accomplished by treating passwords as any other attributes and using Connectors as shown in the previous sections. However, this module provides enhanced security for this critical data. The password intercept module is available only for certain platforms, such as Microsoft Active Directory, IBM Lotus Domino, and RACF®.

When a user attempts to change a password using the traditional tools, this module intercepts password changes before they are completed. While the password change to the target repository is completed with the native methods, the intercepted new password is temporarily stored in a repository such as an LDAP server or an MQ queue. Then Directory Integrator uses an EventHandler to propagate the new password to other repositories that contain user accounts. Because the password is intercepted before it is actually changed, error handling is possible.

Figure 3-22 on page 125 shows what happens when a user changes the Windows Domain password. The password synchronization module hooks an exit provided by the Windows Operating System to intercept and validate password changes. The module stores the two-way-encrypted new password in the LDAP directory in the *ibmDIKey* attribute for the user's entry. If no entry for the user exists in the container, one will be created. The LDAP Changelog Event Handler listens to the TDS Changelog and starts an AssemblyLine when a change notification is received.

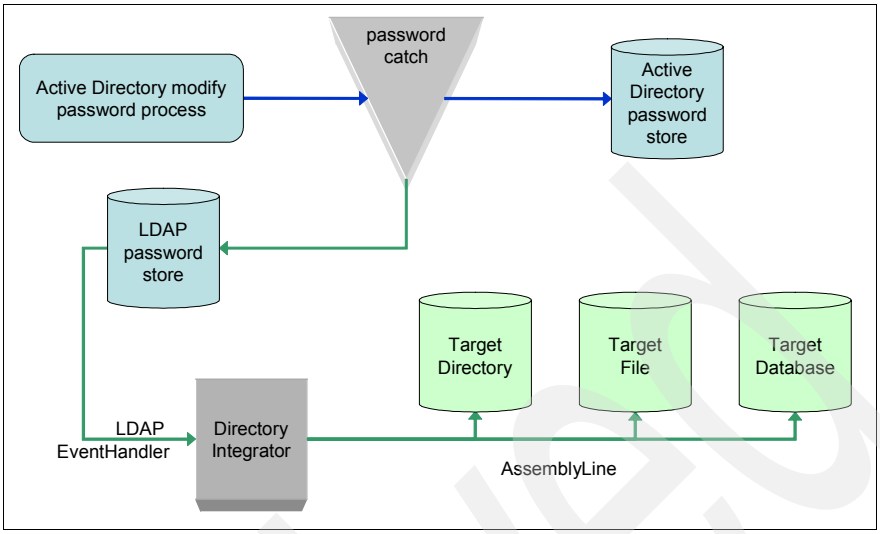


Figure 3-22 Password interception with Active Directory

Security is a strong point of password synchronization modules: The password interceptor encrypts the new password with a two-way algorithm before sending it to the data store. Furthermore, SSL can be added to this communication. In general, IBM Tivoli Directory Integrator provides high security in this module and in all of its parts. In IBM Tivoli Directory Integrator multiple password synchronization plugins can share the same MQ queues simplifying setup and maintenance of multi-domain password synchronization solutions.

3.5.4 Security capability

Directory Integrator supports distributed environments through a wide range of communication modes, including TCP/IP, HTTP, LDAP, JDBC, and Java Message Service (JMS)/message queuing (MQ). SSL and other encryption mechanisms can be added to any of these methods to secure the information flow. Additionally, the Tivoli Directory Integrator server and the AMC are ssl-enabled by default, so communications between the browser and AMC or between the Tivoli Directory Integrator Configuration Editor and Tivoli Directory Integrator server are encrypted. The Java Cryptography Extension (JCE) opens a wide range of security capabilities, such as encrypting information in communications and storage, X.509 certificate, and key management to integrate with PKI efforts in the enterprise.

The AMC supports client certificate authentication and access rights to the Directory Integrator configuration can be defined per user. The configuration file

can optionally be encrypted by Directory Integrator server using server certificate. Config Editor accesses such configurations in *remote mode*.

In the previous sections we introduced the base components and showed that a wide range of data sources is supported. We just saw that communication between different systems can be encrypted. With these elements, hundreds of different solutions can be set up to fit different requirements. In the following section we show some architectural concepts and some examples.

3.5.5 Physical architecture

IBM Tivoli Directory Integrator can be presented through a number of use cases that can illustrate the technical capabilities and some of the solutions that can be architected, but we cannot show all possible architectures with all of the different data sources and data flows. So we introduce some general considerations about the use of an enterprise directory and some basic structures of data flow, not as a comprehensive list, but as frameworks or some mental structures to the creative mind for further development.

Combination with an enterprise directory

There are two major Metadirectory models or approaches to integrating existing enterprise data stores and building an authoritative source for identity information that exist:

- Metaview, which introduce one main central directory store where all data is aggregated and then synchronize and publish data from there back to all other authoritative repositories.
- Point-to-Point synchronization, to avoid the central repository and configure events driven automatic data flow and reconciliation between the repositories, based on business rules and technical requirements.

Metadirectories are often used to accomplish the following goals:

- Create a single enterprise view of users from attributes stored in network services.
- Enforce business rules that define the authoritative source for attribute values.
- Handle naming and schema discrepancies.
- Provide data synchronization services between information sources.
- Enable network and security administrators to manage large, complex networks.
- Simplify the management of user access to corporate resources.

As the foundation for a Metadirectory solution, IBM Tivoli Directory Integrator supports both solutions and provides a means of managing information that is stored in multiple directories. It provides Connectors for collecting information from many operating system and application specific sources and services, as well as for integrating the data into a unified namespace. It can provide a central enterprise directory, as well as integrate distributed directories directly.

By design IBM Tivoli Directory Integrator seems especially suited for the second approach. As a Metadirectory, it extends the directory with services for managing information that is stored in multiple directories. It acts as the hub for making changes between the disparate systems, and it has a number of facilities that enable it to act as the agent for change on these disparate systems. A scenario based on this architecture is shown in Figure 3-14 on page 99. The important design decision is on the authoritative data repository; after that it is a matter of defining the data flows for each AssemblyLine.

There are two possibilities for the implementation of a centralized enterprise directory. The architecture can have one directory with different authoritative data sources for different identity information as shown in Figure 3-23 on page 128, or you can define your central directory as the authoritative data source. In this case, all of the data flows have to be configured in a way such that the central directory server is the prime source for all identity information within the integrated environment.

For our scenario depicted in Figure 3-23 on page 128 we would have to change the arrows to allow data flows only from the enterprise directory to the other repositories. This means that data is essentially managed only on one directory server, and then IBM Tivoli Directory Integrator propagates any changes to the other repositories.

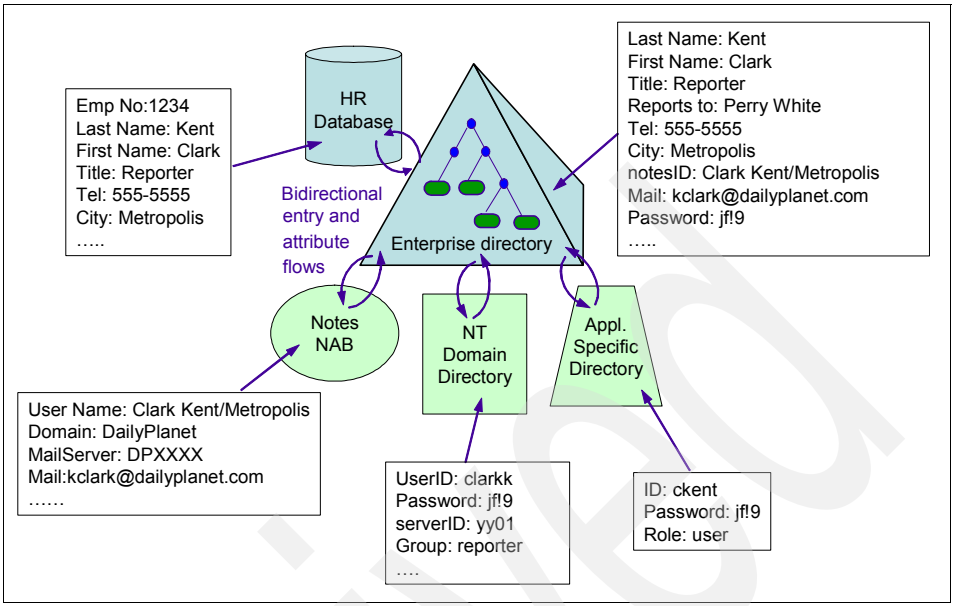


Figure 3-23 Scenario with an enterprise directory

The choice between the solutions depends on the company requirements and structures. There are no technical issues that favor one or the other approach. Mainly it is a matter of choosing the authoritative source for your identity information and considering management, security, privacy, economic, and risk issues.

Regardless of the choice you make, the basic element for identity data integration is data flow. To architect an integrated and reliable identity infrastructure, several data flows must be implemented. Therefore in typical solution design you must determine:

- How does information flow between systems?
- When does information flow between systems?
- What data and schema transformations are required?

In the next section we discuss different topologies available for data flows.

Base topologies

In this section we present some topologies that can be used to architect more complex solutions. For every topology, we identify a data source, a flow, and a destination. In the following examples, each element is drawn in separate boxes. This is just a logical separation. From the physical point of view some of these elements might reside on the same machine. For instance, it is quite common to

place a IBM Tivoli Directory Integrator server on the same machine as its data source. The decision of whether to use different servers depends only on performance and availability.

One-to-one

We begin with the easiest scenario shown in Figure 3-24. Data exists in a file that must be synchronized, transformed, and maintained in a directory. This file could be updated regularly by an HR application or other enterprise systems.

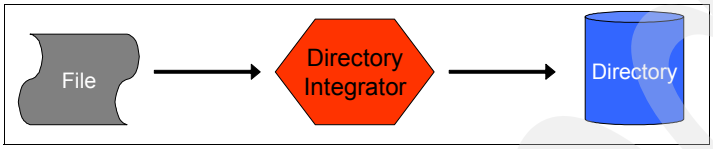


Figure 3-24 One-to-one integration

A wide range of file formats can be accommodated for the input file. The selection on the file format is defined in the input connector, mostly configured in iterator mode. Different ways are available to manipulate and filter the input data stream, such as using the parser or different scripting methods. A separate output connector is established to the directory. IBM Tivoli Directory Integrator discovers the attributes in the file and enables mapping to attributes in the directory as well as applying transformation rules to modify the content of the incoming data.

The file can be read at regular intervals, or read whenever IBM Tivoli Directory Integrator discovers that it's available. The outside application may also trigger IBM Tivoli Directory Integrator to read the file at its own leisure.

Many-to-one

The second scenario is shown in Figure 3-25. Data exists in multiple related systems that have to be synchronized, transformed, and maintained in a directory. Different attributes of data must be joined before an update to the directory can take place.

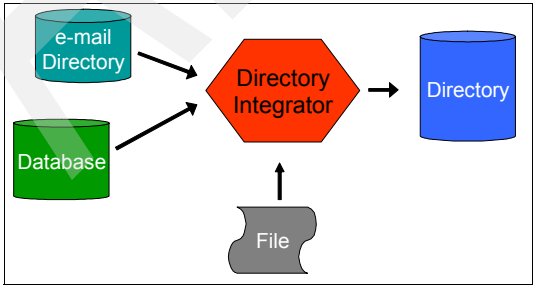


Figure 3-25 Many-to-one integration

Connections are established to each data source using input connectors. Schemas in databases are automatically detected. Rules may be created that describe how attributes from one source are used with attributes from other systems to create the desired results. Information from the data sources can be combined in any way and mapped to the directory. Administrators can select the authoritative source for each piece of information. Data from one system may be used to look up information in another.

IBM Tivoli Directory Integrator can detect changes in real-time within certain directories, allowing immediate update of other connected systems. Connections may be configured to lookup only data that has been modified within a certain time frame, or data sets that conform to a specific search criteria.

One-to-many

A one-to-many scenario is the opposite of that described in the previous example. Information updated in one source is propagated to many destinations. IBM Tivoli Directory Integrator can perform exactly the same write, update, delete, and create modifications on all connected systems as it does for directories. The rules are simply adapted for the context. Now all systems can share the common authoritative data set.

In this third scenario, presented in Figure 3-26 on page 131, we introduce bidirectional flows. Bidirectional flows can be configured such that there is either only one authoritative data source for each piece of information or concurrent authoritative sources for the same data. In the second case the data in the directory is provisioned from multiple connected systems as well as from possible modifications done by applications connected to the directory. The connected systems could have great interest in this data, especially when IBM Tivoli Directory Integrator ensures that they always operate on the correct information by updating them whenever the authoritative data changes.

By configuring the connectors, using hooks and scripting, administrators can apply rules to define and monitor the flows. However, we recommend being careful with multiple data sources for the same piece of information. A good idea is to have only one point where specific data can be modified. This is not a technical issue, because IBM Tivoli Directory Integrator easily allows multiple data sources. It is a matter of implementing clear processes and data flows. On the other hand, it is common and often advisable to have sources for specific data on different systems.

For example, in Figure 3-26, users could modify their e-mail address or preferences only in the e-mail database, while they could change their password only with an application that directly interacts on the Directory.

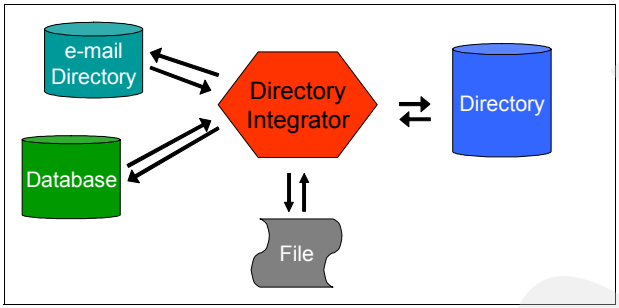


Figure 3-26 One-to-many integration

Other data resources

There are many reasons why data flows through channels such as message queuing, HTTP, e-mail, FTP and Web Services. Data might need to pass through firewalls that block protocols like LDAP and database access. Security, high-availability, transactions control and desire for asynchronous or synchronous data transfer are other reasons.

It's important to understand that IBM Tivoli directory Integrator can both send and receive with these mechanisms. This creates a wide scope of solution opportunities, too wide to describe in simple use cases. Some examples are illustrated in Figure 3-27.

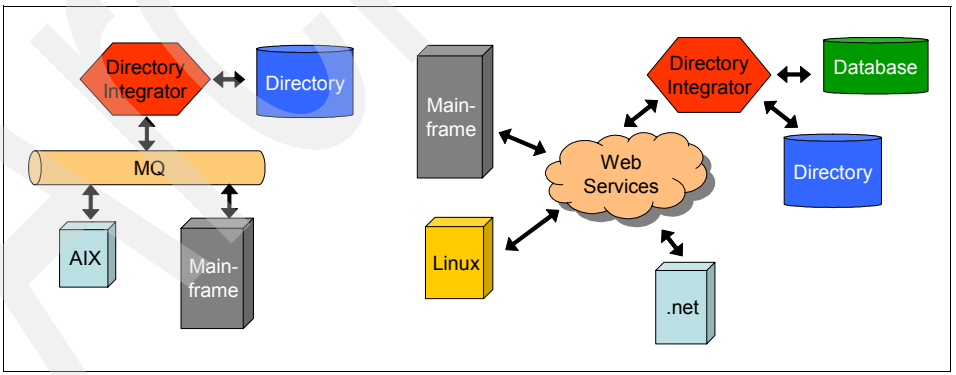


Figure 3-27 Other data sources integration

Multiple servers

In the scenarios shown so far, there is only one IBM Tivoli Directory Integrator server. In this section we present some topologies with multiple server instances.

Distributed

In a distributed architecture, a single point of integration is often undesirable, for reasons such as distance, financial, security, availability or governance.

All of the mechanisms described previously, such as IP, HTTP, Web Services, e-mail, MQ and others can be used to communicate between instances of IBM Tivoli Directory Integrator.

In Figure 3-28 the arrows indicate use of such communications mechanisms in two examples. In first example the input stream is too fast compared to the business rules that IBM Tivoli directory Integrator has to execute and multiple instances can operate of a queue. In the second example two-way architecture propagates updates in the directory to the rest of the enterprise and consolidates local modifications back to the central directory.

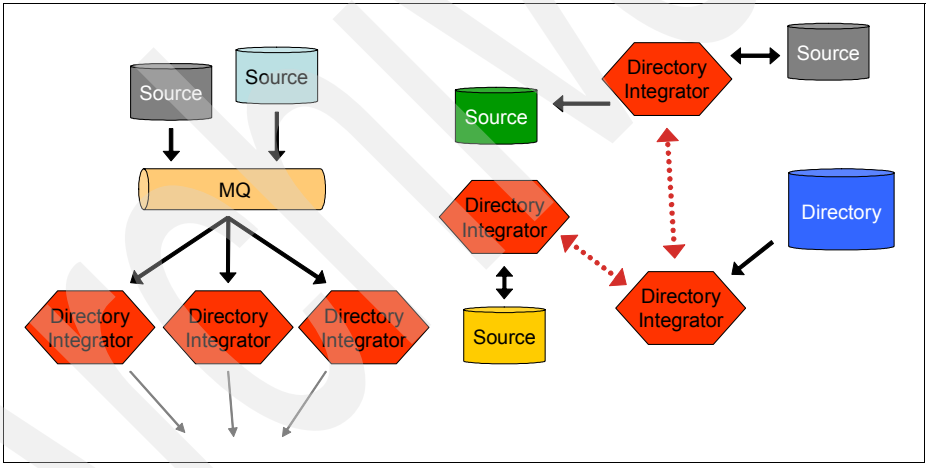


Figure 3-28 Distributed integration

Federated

While similar to the distributed scenario, federated implies that control and management is not entirely centralized. This could be business units or entities that cooperate, but want to retain local control over how and what information is shared with others.

By sharing certain parts of the Directory Integrator configuration, Directory Integrator servers have access to shared transports, formats and business rules

Example scenario shown in Figure 3-29 could be that different business units want to retain local control over information shared with others. Local configuration allows administrators to set restrictions on the data sets that are exposed, the attributes that are sent and received, as well as any local transformation rules that need to be applied to the data going to or coming from the other participants.

If a company is spread across multiple sites, it could be beneficial to have an IBM Tivoli Directory Integrator server in each location and then to have data flows only between these servers.

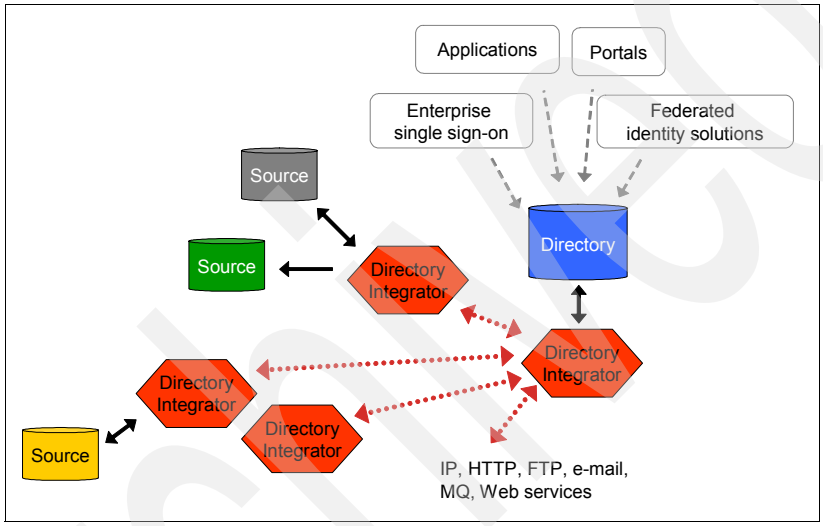


Figure 3-29 Federated integration

The main message in this section is that IBM Tivoli Directory Integrator enables you to use any topology and different transport mechanisms to integrate data stored in various formats on multiple disparate systems.

In the following section we introduce another level of complexity by using multiple servers to implement high availability and increase performance.

3.5.6 Availability and scalability

High availability means that the IT service is continuously available to the customer, as there is little downtime and rapid service recovery. The achieved availability can be indicated by metrics. The availability of the service depends on the following:

- Complexity of the infrastructure architecture
- Reliability of the components
- Ability to respond quickly and effectively to fault

There are several high availability mechanisms inside IBM Tivoli Directory Integrator on various levels from Connectors and AssemblyLines to Server itself. Let's take a brief look at some of them starting from lower level.

Automatic connection reconnect

AssemblyLines need to access remote servers. Ideally, those remote servers should be online and available for the entire time the AssemblyLine is running. In the real world, however, server and network failures are common.

IBM Tivoli Directory Integrator has an automatic reconnect feature. This is sufficient for short term outages, where the AssemblyLine can just try to reconnect until it succeeds. You do this in the Connector's Reconnect sub-tab as shown in Figure 3-30.

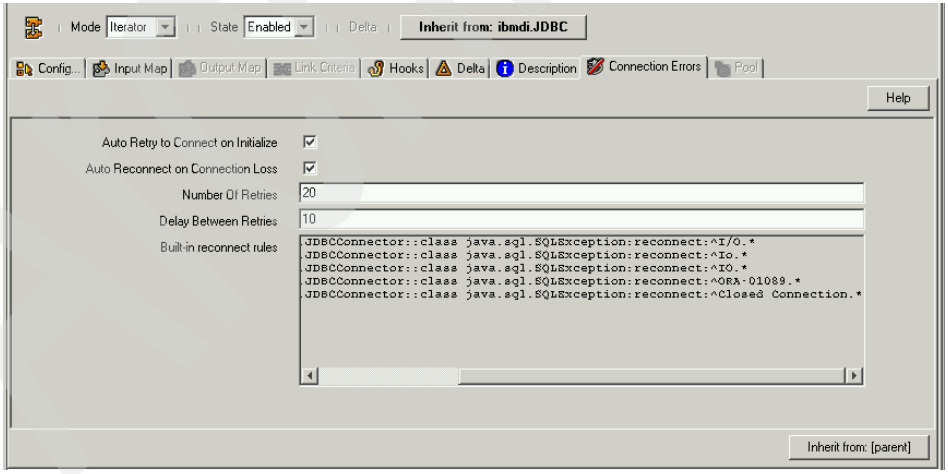


Figure 3-30 Automatic connection reconnect

The parameters you need to provide are:

Auto Retry to Connect on Initialize
: Enable reconnection, even in the first connection try.

Auto Reconnect on Connection Loss
: Enable reconnection during connector operation.

Number of retries
: The number of times the Connector will try to re-establish the Connection, after it fails. The default is 1. When the number of retries is exceeded, an exception is thrown.

Delay between retries
: The number of seconds to wait (in seconds) between successive retry attempts. The default is 10 seconds.

Built-in Reconnection Rules
: Display the events that trigger a reconnection attempt. Each connector implementation has its own reconnection rules, if any.

This also means that AssemblyLine Connectors have a *.reconnect()* method that can be called from script as needed.

If a connection is lost, control passes to the *On Connection Failure* Hook if enabled. This Hook is available in all Connector Modes. Once the Hook completes (or skipped if not enabled) the system then checks if Auto Reconnect has been enabled for this Connector. If it is, then this feature is invoked, otherwise control is passed to the Error Hooks as normal.

Typical use of the *`On Connection Failure`* Hook is to write some message to the log, or even change Connector parameters — for example, pointing it some backup data source. However, since reconnect may not be implemented for a Connector you are using, you can simulate reconnect yourself in the *`On Connection Failure`* Hook by terminating and then re-initializing the Connector with script code.

Note: If you do not want the Connector to Auto Reconnect after invoking the On Connection Failure Hook, you must either disable Auto Reconnect or redirect flow by throwing an exception (with calls like system.retryEntry() or system.skipEntry()) or by stopping the AL itself with system.abortAssemblyLine (message).

AMC Action Manager

The *Action Manager* (AM) is a standalone Java(TM) application that allows you to monitor multiple Tivoli Directory Integrator Servers and AssemblyLine execution using user-defined rules, triggering conditions and actions defined in

the AMC. The *Administration and Monitoring Console* (AMC) has an AM Configuration panel that allows users to configure various *Action Manager rules*.

A rule is a combination of a *Trigger type* and a set of associated actions. A rule specifies that when a Triggering condition is detected, then the associated set of actions must be executed. The following describes the various trigger types available in AMC:

- No trigger

  A rule with this triggering type has no triggering condition, and hence will never get triggered by itself. The only way this rule can be executed is if some other rule executes this rule.

- On AssemblyLine termination

  A rule with this triggering type will get triggered when the Action Manager receives an AssemblyLine termination event for this particular AssemblyLine.

- Time since last execution

  A rule with this triggering type will get triggered when the Action Manager detects that the specified AssemblyLine has not run for the specified period.

- On query AssemblyLine result

  A rule with this triggering type is triggered when the last *work* entry of the specified AssemblyLine, contains the specified *Attribute* matching the given *condition* and *value*. This condition will be checked only when the Action Manager receives a Stop AssemblyLine event.

- On server API failure

  A rule with this triggering type will be triggered when the Action Manager is unable to connect to the remote server using the Server API. No details required.

- On received Event

  A rule with this triggering type will be triggered when the Action Manager receives an event that satisfies the criteria mentioned.

- On Property

  A rule with this triggering type will get triggered when the specified property meets the specified condition. The Action Manager periodically checks for this property.

When a rule gets triggered, the Actions associated with the rule are executed by the Action Manager sequentially. Following are the various types of Actions that are available in AMC:

- Start AssemblyLine

  This action starts the specified AssemblyLine of the specified config file on the specified Tivoli Directory Integrator server. The Config field should mention the complete path of the configuration on the remote server.

- Stop AssemblyLine

  This action stops the specified AssemblyLine of the specified configuration on the specified Tivoli Directory Integrator Server.

- Enable/Disable AM Rule

  This action will Enable or Disable the chosen AM rule.

- Execute AM Rule

  This action will cause the execution of the specified rule, which will in-turn imply execution of all the actions specified in that particular rule.

- Notify Event

  This action will cause the Action Manager to emit an event with the specified details to the Server associated with the current config view.

- Modify Property

  This action will cause the Action Manager to modify the selected property based on the specified operation.

- Copy Property Value

  This action will cause the Action Manager to copy the value of the Source property to the Destination property.

- Write to Log

  This action will cause a log of the specified Severity/Message/Description to be logged into the Action Manager logs and the AMC database. The same log is shown in the AM Results table. It is advised to always have at least one Log action (containing descriptive text) in every rule.

Rules that are configured for Config views in AMC, are stored in AMC's Cloudscape™ Database. When the Action Manager is run, it connects to the AMC database in network mode, reads the Action Manager-related tables, and creates threads in memory for every AM rule specified. Each of these threads listens and polls for its respective triggering conditions. The moment any thread detects the occurrence of its respective triggering condition, it queries the database for the set of actions associated with the rule, and executes them sequentially.

The Action Manager runs the following threads in addition to the rule threads that are listening for trigger conditions:

- HealthAssemblyLine

  The HealthAssemblyLine thread periodically triggers the HealthAssemblyLines for querying the status of the solutions, and logging the status back into the AMC database.

- ServerStatusListener

  The ServerStatusListener thread is created for every server registered with AMC. This server checks for the server accessibility. If the server has become inaccessible, all rules threads created for the server are terminated (except for those with triggering type *On Server API failure*). Similarly, if the server becomes accessible, rule threads are created for any rules associated with this server.

- ConfigLoadReloadListener

  The ConfigLoadReloadListener thread is created for every running server registered with AMC. It is registered to the remote server for any config load or unload events. Rule threads are appropriately terminated, created, or refreshed depending on the config event.

- ServerModificationListener

  The ServerModificationListener thread checks for any updates to the set of servers registered in AMC. Depending on the type of change (added, removed, and so forth), rule threads are terminated, created, or refreshed.

- ActionManagerStatusUpdate

  The ActionManagerStatusUpdate thread updates AMC on whether the Action Manager is currently running or not.

- DatabaseModificationListener

  This database listener thread continuously monitors addition, modification, or deletion of rules. Whenever any changes in the rules are detected, the AM threads are added and recreated appropriately at runtime.

The Action Manager also updates the AMC database with its run details. Whenever an Action Manager rule is triggered, Action Manager logs an entry into the AMC database, registering the rule name that got triggered, and the triggering time. Also, if any AM Log action is configured for the AM rule, then that also gets logged into the AMC database. These database entries show the appropriate status in Monitor Panels of AMC.

Automatic high availability

The basic concept of high availability is to have at least two servers capable of performing the same job and a failover mechanism to switch from one server to the other if one of the servers fails.

IBM Tivoli Directory Integrator does not provide such failover mechanism out-of-the-box. Therefore, one way to provide automatic high availability is to implement architecture as shown on Figure 3-31, where one IBM Tivoli Directory Integrator Server instance is configured to watch the other just-in-case and can take over if the second one fail to respond.

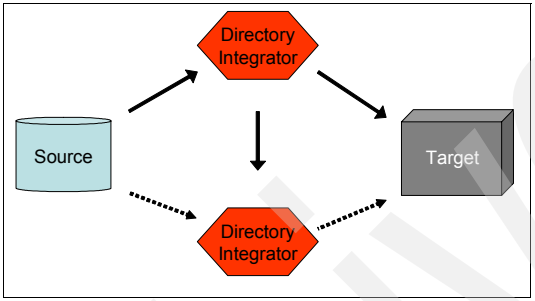


Figure 3-31 Just-in-case high availability

The other possible way of high available automatic fail over mechanism is to install the server in a cluster environment such as HACMP for AIX as shown on Figure 3-32.

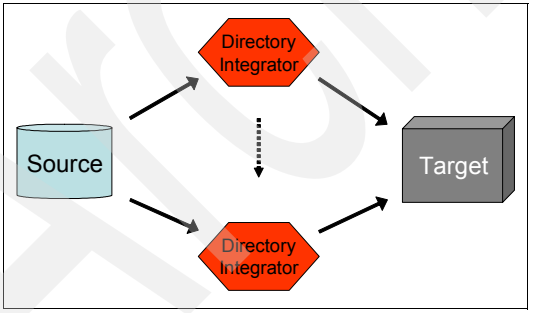


Figure 3-32 Clustering

However, remember that all AssemblyLine definitions and configurations are stored within one highly structured XML file called Config. Therefore, if one server fails, it is sufficient to start a separate server with the same Config file in order to continue the service. IBM Tivoli Directory Integrator's main goal is to perform data integration, not real-time services. This means that a short period of

unavailability (for example, for maintenance reasons) can be tolerated in most cases.

A failover mechanism must be configured between the two servers, depending on functional requirements of the data integration environment.

Scalability is a strong feature of IBM Tivoli Directory Integrator. There is virtually no limit to the number of servers that can be added. As it was already shown on Figure 3-28 on page 132, different servers can work on different data flows or on different data of the same data flow.

Considering the AssemblyLine mechanisms, no additional effort is required to integrate multiple servers. Each AssemblyLine is designed to work on different data. Different AssemblyLines integrate different data sources regardless of whether these AssemblyLines reside on the same server or on multiple servers.

AssemblyLine Pool

With AssemblyLine Pool you can build high performance solutions that won't incur a thread and connection cost for each processed event. You can configure Pool options from the *Show Dialog* button next to *Define ALPool Options* on the Config tab of AssemblyLine as shown on Figure 3-33

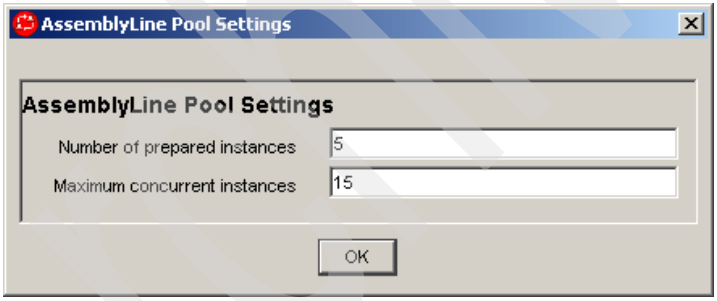


Figure 3-33 AL Pool

The parameters you need to provide are:

Number of prepared instances How many instances of the Flow part of this AL to instantiate, power up and then keep in the Pool, ready for use.

Maximum concurrent instances What is the maximum number of current Flow instances that you want created at any one time.

Note: Pooling is only available if you have a Server Mode Connector in the Feeds section of your AssemblyLine.

See the *IBM Tivoli Directory Integrator: Users Guide* for more ALPool details.

3.5.7 Logging

IBM Tivoli Directory Integrator enables you to customize and size logs and outputs. It relies on log4j as a logging engine. Log4j is a very flexible framework that lets you send your log output to a variety of different destinations, such as files, the Windows EventLog, UNIX Syslog or a combination of these. It is highly configurable and supports many different types of log appenders and can be tuned so it suits most needs. It can be a great help when you want to troubleshoot or debug your solution. In addition to built-in logging, script code can be added in AssemblyLine to log almost any kind of information. If the logging functionality will not suffice, the there are additional tracing facilities.

The log scheme for the server (ibmdisrv) is described by the file log4j.properties and elements of the Config file, while the console window you get when running from the Config Editor (ibmditk) is governed by the parameters set in executetask.properties. Logging for the Config Editor program itself is configured in the file ce-log4j.properties.

Note: Any of the aforementioned properties files can be located in the Solutions Directory, in which case the properties listed in these files override the values in the file in the installation directory.

You can create your own appenders to be used by the log4j logging engine by defining them in the log4j.properties file. Additional log4j compliant drivers are available on the Internet, for example drivers that can log using JMS or JDBC. In order to use those, they need to be installed into the IBM Tivoli Directory Integrator installation jars directory after which appenders can be defined using those additional drivers in log4j.properties.

Configuring the logging of IBM Tivoli Directory Integrator is done globally using the files log4j.properties and/or External Properties or specifically, using the ibmditk tool, for each AssemblyLine, EventHandler or Config File as a whole. Logging for individual AssemblyLines and EventHandlers is applied in addition to any specification done at the Config level.To provide this level of flexibility and customization, the Java Log4J API is used.

All log configuration windows operate in the same way: For each one you can set up one or more log schemes. These are active at the same time, in addition to

whatever defaults are set in the log4j.properties and executetask.properties files. In Figure 3-34 you can see an example of the Syslog scheme, which enables IBM Tivoli Directory Integrator to log on UNIX Syslog.

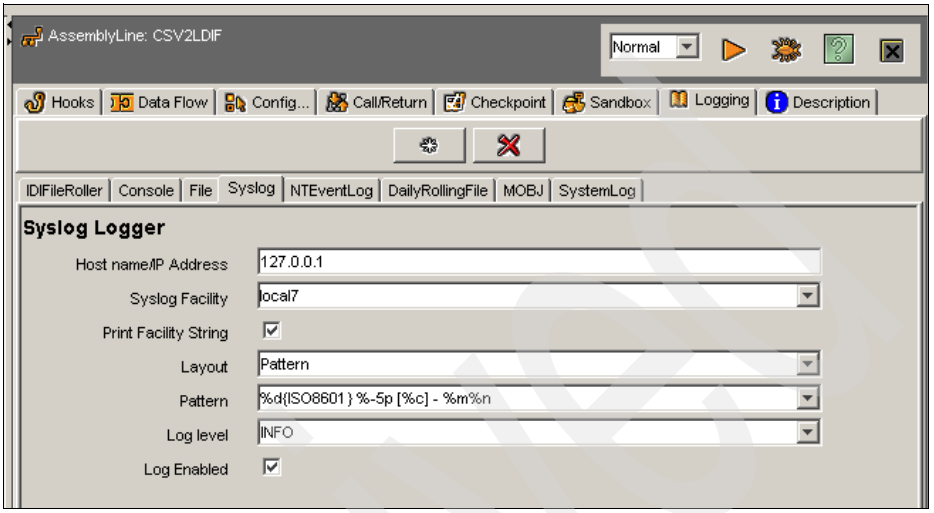


Figure 3-34 Syslog scheme

See the *IBM Tivoli Directory Integrator: Administrator Guide*, for more details on schemes configuration.

Key data is logged from the Directory Integrator engine, from its components (Connectors, Parsers, and so on), as well as from user's scripts. Almost every Connector has a debug parameter called Detailed Log, with which you can turn on and off the Connector's output to the log file. Seven log levels range from ALL to OFF for sizing the output. ALL logs everything. DEBUG, INFO, WARN, ERROR and FATAL have increasing levels of message filtration. Nothing is logged on OFF.

Note: IBM Tivoli Directory Integrator logmsg() calls log on INFO level by default. This means that setting loglevel to WARN or lower silences your logmsg as well as all Detailed Log settings. However, the logmsg() call also has a level parameter that can be used to override the log level for individual logmsg() calls.

In order to augment the IBM Tivoli Directory Integrator built-in logging, you can create your own log messages by adding script code in your AssemblyLine. Different information can be dumped, such as the content of an Object or Attribute, the state of a Connector, or any desired text. This means that you can indicate to the log file or to the console any state of the custom logic of your

AssemblyLines. See the *IBM Tivoli Directory Integrator: Users Guide*, for more logging details and examples.

Note: Errors from Attribute Map Components do not show the name of the Attribute Map Component, only the name of the AssemblyLine, and often (depending on the error), the name of the attribute being mapped. The message will often contain the name of the attribute that is mapped, which should give you a hint as to which Attribute Map it is that fails.

Tombstones

Tivoli Directory Integrator can keep track of configurations or AssemblyLines that have terminated. Thus you can tell when your AssemblyLines last ran, without going into the log of each one.

This is accomplished by Tivoli Directory Integrator's *Tombstone Manager* that creates *tombstones* for each AssemblyLine and configuration as they terminate. They contain exit status and other information that can later be requested through the Server API. This also enables the following:

- A status panel in AMC that displays the status of an entire Tivoli Directory Integrator configuration.
- Functionality within Action Manager to ensure repeated runs of AssemblyLines, for example every 24 hours.
- Provision of status information to Server API clients about AssemblyLines that they run asynchronously.

Debugging

In addition, IBM Tivoli Directory Integrator offers you a Flow Debugger (not to be confused with a script debugger). The Flow Debugger lets you step through your AssemblyLines and examine and change variables and/or run script directly. An example of Flow Debugger usage is shown in Figure 3-35 on page 144.

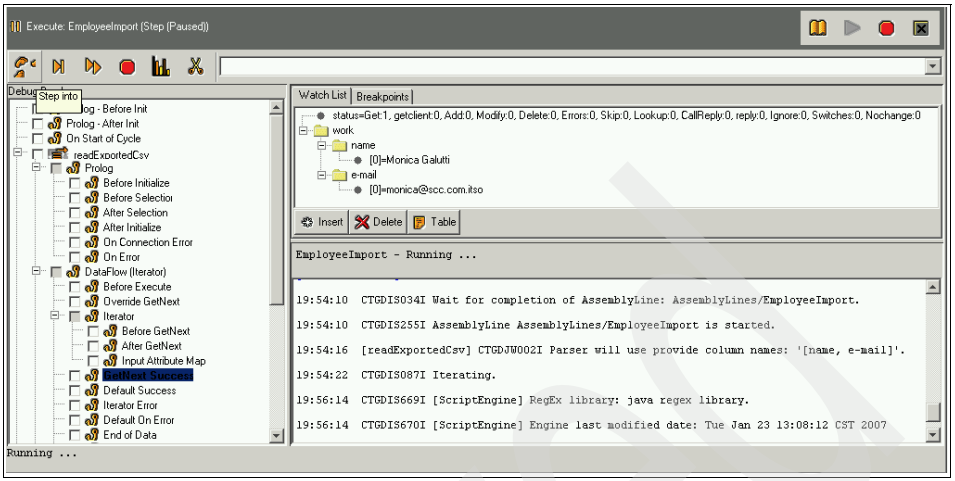


Figure 3-35 Flow Debugger

The debugger is started from the Config Editor by selecting one of the debugging options before clicking the Run button from the AssemblyLine. After the selected task is started, the Debugger pauses, processing at specified breakpoints. It can also be configured to pause at every step. Whenever execution is paused, you can use the Table and Statistics buttons to display information about run a script. There is also an Edit watch list button that offers you the same option, however the resulting watch-list is remembered and evaluated at each breakpoint. One example of a variable you might want to watch is work (the work Entry object). By entering work in the Evaluate dialog, or adding it to your watch-list, you can see work serialized to the Output pane of the debugger.

Note: If you evaluate (or watch) the script task.dumpEntry(work), then the work Entry is dumped to the log output pane instead, just as though you had this code in your solution.

Tracing

In addition to the user-configurable logging functionality described in previous section, IBM Tivoli Directory Integrator is instrumented throughout its code with tracing statements, using the JLOG framework, a logging library similar to log4j, but which is used inside Directory Integrator specifically for tracing and First Failure Data Capture (FFDC). To which extent this becomes visible to you, the user, depends on a number of configuration options in the global configuration file jlog.properties, and the Server command line option -T.

Tracing is done in using JLOG's PDLogger object. PDLogger or the Problem Determination Logger logs messages in Logxml format (a Tivoli standard), which IBM Support understands and for which they have processing tools.

Note: Normally, you should be able to troubleshoot, debug and support your solution using the logging options. However, when you contact IBM Support for whatever reason, they may ask you to change some parameters related to the tracing functionality described here to aid the support process.

See the *IBM Tivoli Directory Integrator: Administrator Guide* for more tracing details, configuration, and parameters.

3.5.8 Administration and monitoring

Config Editor is a program that gives you a graphical interface to create, test, and debug any AssemblyLines with all of the components and the optional scripting. It is an Integrated Development Environment (IDE), introduced in 3.5.3, "Base components" on page 101, used to create a configuration file that describes your solution and is powered by the runtime Server. This configuration is called a Config, hence the name Config Editor.

The Config Editor is started by initiating the ibmditk batch-file or script, which sets up the Java VM environment parameters and then starts the Config Editor. It enables you to work with multiple Configs at the same time. Configs are stored as highly structured XML documents and can be encrypted. When you start the Config Editor, either from your system's launch interface or from the command line with the ibmditk command, you will see the Main Panel.

In the default layout the left navigation pane provides a tree view of the current configuration, as well as all the current AssemblyLines, Connectors, and so forth as shown in Figure 3-36 on page 146. AssemblyLines can be created easily by selecting components. The Attributes definition in the connected elements is automatically discovered and mapping can be done simply by dragging or renaming attributes.

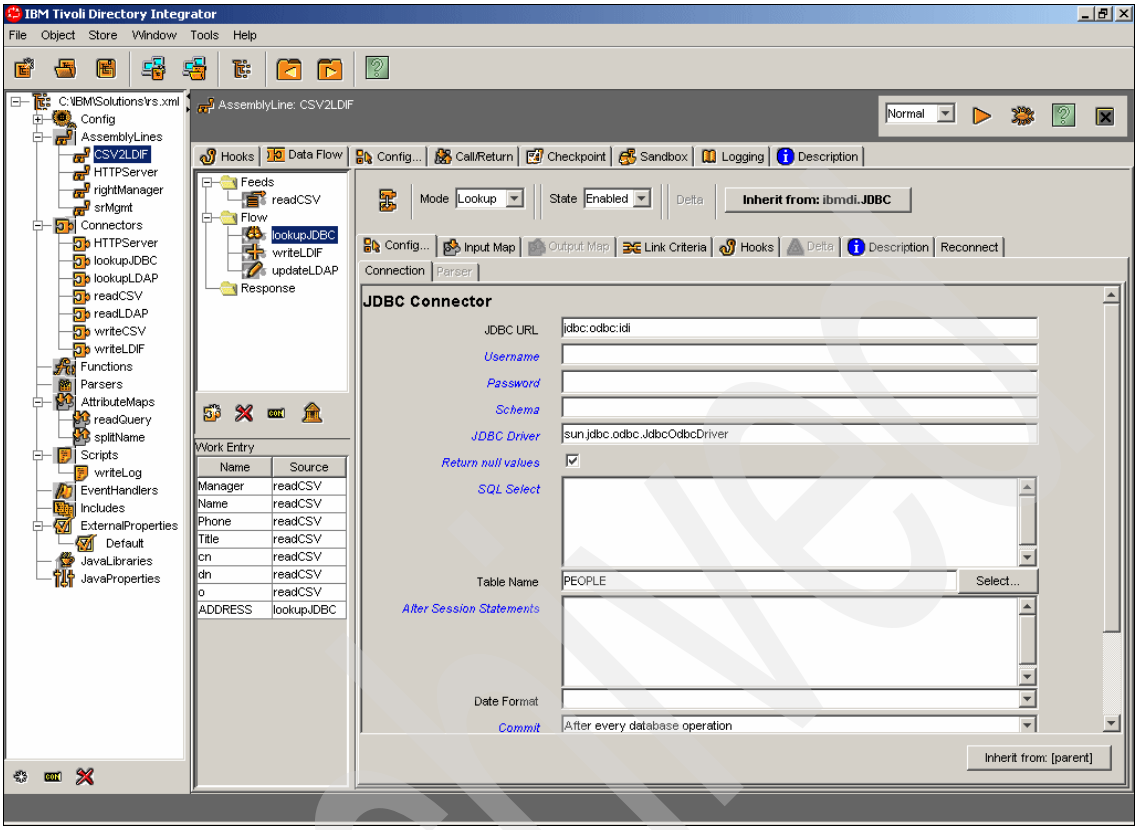


Figure 3-36 Config editor main panel

See the *IBM Tivoli Directory Integrator: Users Guide*, for more details on the Config Editor.

When the AssemblyLines are ready and the integration solution is deployed, administration and monitoring can be performed.

After the integration solution is in maintenance mode, operators need to be able to run AssemblyLines manually. One option is to give operators access to the Config Editor. However, since operators should not modify AssemblyLines, this option violates the principle of least privilege. Another possibility is to let operators run AssemblyLines from the command line. However, unless they need shell access for a different reason, this also violates the principle of least privilege. Tivoli Directory Integrator bundles a Web-based Administration and Monitoring Application (AMC). The AMC can be used to remotely start, stop, and manage Tivoli Directory Integrator Configs and AssemblyLines, which allows

operators to only perform the actions they are allowed to do, and to do so from a user friendly Web browser environment.

Note: The principle of least privilege states that users should only be given those permissions they need to do their jobs. For example, operators who do not need to change IBM Tivoli Directory Integrator AssemblyLines should not be allowed to do it.

AMC is a Java Web-based application that uses the Tivoli Directory Integrator Remote Server API. In addition to AssemblyLines monitoring, Config Administration, Property Stores Administration, Log files cleanup, Console users and groups management, you may also set up connections to multiple IBM Tivoli Directory Integrator server instances and configuration files running on them.

AMC communicates with IBM Tivoli Directory Integrator servers over SSL using the Java Security Extensions. It is pre-configured to work with the server that it is bundled with. In order to use AMC with servers that use other certificates than the one they were shipped with, the server certificates need to be added to the AMC truststore, and the AMCs certificate needs to be added to the server truststores.

An important concept introduced in the AMC is the *Config View*. The Config View gives users access to information in a configuration file without granting them the ability to edit the configuration file directly. Administrators can use a Config View to filter a configuration file for specific information such as properties and AssemblyLines so that only certain information within the configuration file is displayed.

You can create multiple Config Views for each Config. Each view can expose different information contained in the configuration file, while also allowing you to hide unnecessary information from the user.

AMC permissions are assigned per Config. This enables IBM Tivoli Directory Integrator to enforce a separation of roles even when the same server is used for multiple purposes in the organization. For example, a server might be used to synchronize both user accounts and office supply information. If you put all the AssemblyLines related to users in one Config and all the AssemblyLines related to office supplies in another, then operators can have permissions to one but not the other.

There are three permission levels in AMC:

Read This means read-only permission. The user cannot change anything or run anything. This level is useful for auditors and operators in training.

Execute This level allows users to execute AssemblyLines and EventHandlers, and view and delete the resulting logs. However, users with execute permissions are not allowed to modify or delete any components or component properties. This permission level is for operators.

Admin This level allows full control of IBM Tivoli Directory Integrator, similar to the control available through the Config Editor.

See the *IBM Tivoli Directory Integrator: Administrator Guide* for more details about AMC files, setup, and configuration.

3.6 Conclusions

In this chapter we discussed the need for a centralized user repository with a single point of administration. Then we introduced the concepts of the directory server and LDAP. LDAP servers are a solution that fit these requirements perfectly.

However, in complex organizations it is sometimes very difficult to consolidate all user definitions in only one repository. This is because some pre-existing applications might be hard to migrate using a single LDAP server. Therefore it might be necessary to maintain multiple repositories, which can be directory servers, databases, flat files, or other. In this kind of environment it is necessary to synchronize these disparate data sources in order to have a consistent identity infrastructure.

We examined virtual directories and meta directories, two technologies for integrating data sources across the enterprise. We examined the benefits and drawbacks to each approach.

We also introduced IBM Tivoli Directory Integrator, a powerful tool to integrate and reconcile data across multiple repositories on different platforms. This product focuses on data rather than on users and it solves the complex integration challenge by breaking it into separated, modular, and scalable pieces.

IBM Tivoli Directory Integrator enables you to create a consistent infrastructure of enterprise identity data, while permitting local administrators to manage users on each platform and environment with their traditional tools.

Chapter 21, “Synchronizing the enterprise” on page 633, takes a best-practice look at different real-world scenarios and describes the solutions that are based on a mix of Identity Manager, Directory Integrator, and Access Manager.

4

Single sign-on technologies

The term single sign-on (SSO) has been bandied about for so long that it has lost much of its initial power. SSO held such power because of its promise to relieve companies of the heavy information technology costs required to maintain security in their information technology enterprise. Despite the best efforts of software vendors, including IBM with its Global Sign-on and Distributed Computing Environment technologies, the goal of each user logging into their computer once and securely accessing all corporate services remained elusive.

When technology failed to live up to consumer expectations, the software vendors decided to lower the expectations by changing the terminology. Single sign-on was dubbed *reduced sign-on* or *simplified sign-on*. While this may have reduced consumer expectations, the fact remained that companies were still investing heavily in supporting the user community, which had to keep track of many login names and passwords.

IBM takes a divide and conquer approach to this intractable problem and addressed different classes of SSO with different technologies. The three classes of SSO are:

- *Web single sign-on*
- *Desktop single sign-on*
- *Federated single sign-on*.

Addressing each of these separately is technically feasible and allows for the realization of true single sign-on.

By surveying the different types of SSO and the benefits of each, you will be in a good position to clearly articulate your company's SSO requirements and to identify a solution that can deliver a full range of SSO capabilities.

4.1 SSO delivers multiple business benefits

To be an on demand business, a company frequently requires SSO capabilities. By providing users with the ability to log in once across the applications and operating systems that they need to access, a business drives both quantifiable and qualitative benefits, including:

- Reduced administration costs

  When users must log in multiple times, they are more likely to forget passwords, which in turn leads to greater Help Desk costs. SSO can significantly reduce these calls and their resulting costs.

- Greater user productivity and experience

  SSO allows users to access business systems faster, which enables them to get more done. And users who can sign in once feel better about their transaction experience than users who must log in multiple times with many different IDs and passwords.

- Faster application deployment

  When companies deploy a superior SSO and security system that allows application developers to call out to external security services, security no longer has to be coded into each application. As a result, a company can get new applications to market quickly, and can later update application business logic and enhance security much more efficiently.

The benefits of SSO grow as it is applied against an expanded pool of IT environments. As computing models have evolved from distributed client/server systems to Web-based applications—and now even to federated SSO configurations often involving emerging standards such as Security Assertion Markup Language (SAML), Liberty Alliance, and Web Services Federation Language (WS-Federation)—businesses are able to realize increasingly significant value from SSO solutions particular to each model.

4.2 Three classes of single sign-on

Let us consider SSO in the context of the evolution of computing models as shown in Figure 4-1 on page 151.

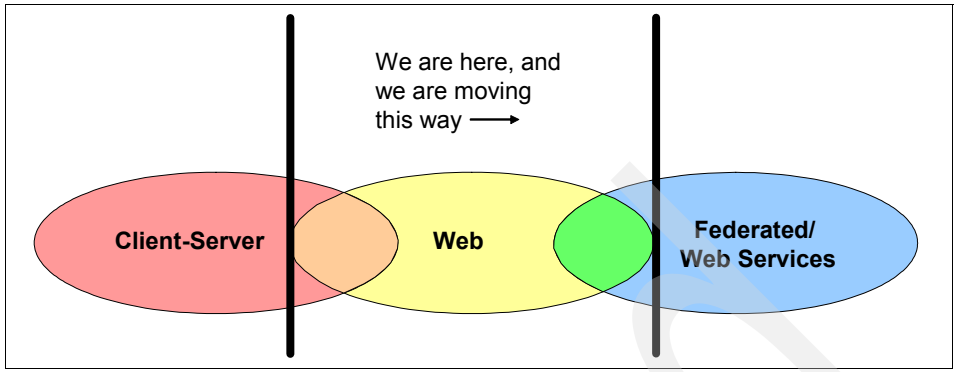


Figure 4-1 Evolution of computing models

As the computing models evolve from the insular client-server model to the open Web services model, the importance of security increases dramatically. The following table (see Table 4-1) demonstrates how this evolution is impacting user authentication.

Table 4-1 Single sign-on models and their authentication characteristics

Model	Network exposure	Communication protocol	Authentication
Client-Server	Private	Proprietary	Authenticates user against an application-specific repository. User population contained within the enterprise.
Web	Private/Public	Standards-based	Authenticates user against an application-specific or enterprise repository. User population expands to the Internet.
Federated/ Web Services	Public	Standards-based	Authenticates user against enterprise repository. Also authenticates users and other network entities (for example, Web services) originating from foreign companies. User population expands to the Internet and other enterprises.

The *client-server model* achieves a certain level of security through the fact that it operates within the corporate network and communicates over a propriety

protocol. However, being a network-based architecture, the client-server model requires client authentication. In this model, each application tends to have its own user repository. This requires users to keep track of separate accounts for separate applications.

The *Web model* is just a special case of the client-server model that uses a standard client and communications protocol (HTTP/S). Companies find this model more cost effective, since only one client needs to be deployed to the corporate desktops. Many traditional Web applications were developed (like the client-server model) using their own user repository. In addition to having the same sign-on problem as the client-server model, the Web model compounds the problem by exposing corporate applications directly to the customers through the Internet. This means that companies face a large increase in the number of users needed to be supported. Also, these users are not the traditional corporate users, but rather customers who the company must vet before assigning accounts.

The emergence of the Web as the platform of choice for corporate applications and the exposure of the corporate applications to the end-users created the opportunity for services to be linked between corporations over the Internet. For example, a corporate Web portal may link off to the health benefits provider and the financial services partner. These links lead to an external Web site that requires authentication. Thus, the user is once again faced with additional account data to manage. The *federated model* requires identity information to be carried securely over the Internet so that users may consume services at various companies.

Each of the three computing models have single sign-on requirements and IBM applies different technologies to meet each of the SSO requirements. These techologies are implemented in the following products:

- Tivoli Access Manager for Enterprise Single Sign-on

  Addresses the desktop SSO problem by deploying an agent on the desktop, which intercepts authentication requests by applications and automatically fills in the login data with credentials stored on the local machine.

- Tivoli Access Manager for e-business

  Addresses the Web SSO problem by placing a reverse Web proxy in front of the enterprise Web applications. Tivoli Access Manager user accounts are stored in an enterprise directory and users need only authenticate to the Tivoli Access Manager server in order to access all of the existing Web applications configured behind the reverse proxy.

- Tivoli Federated Identity Manager and Tivoli Federated Identity Manager Business Gateway

  Address the federated SSO problem by implementing all of the industry standard federated SSO protocols. These are SAML (all versions), Liberty ID FF (all versions), and WS-Federation. It supports arbitrary identity transformations based on XSLT so that credentials can be converted to a format compatible with the local environment.

Figure 4-2 shows a logical diagram depicting how these products fit together in a solution that addresses all three types of the SSO.

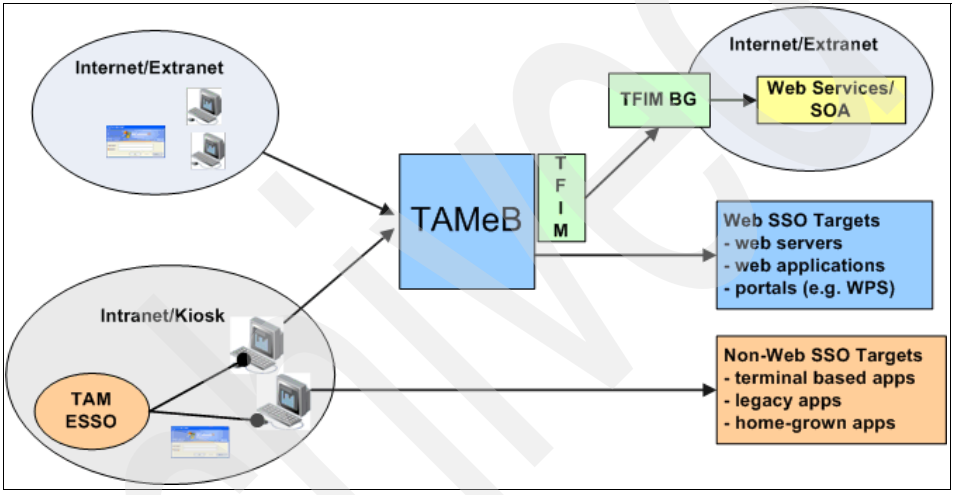


Figure 4-2 IBM Tivoli SSO technologies

The rest of this chapter discusses these SSO technologies in more detail.

4.3 Desktop single sign-on

Although few, if any, of the more modern computing solutions being developed today use the client/server model, many existing legacy client/server applications can still benefit from Desktop SSO. While many companies successfully addressed the SSO problem for Web applications, end users still must log into their Windows desktop, log into their mail client, log into corporate chat applications, log into human resources systems, and the list goes on. A complete SSO solution must address desktop SSO.

Tivoli Access Manager for Enterprise Single Sign-On is designed to be an easy-to-deploy solution to automate user authentication to desktop applications. It provides single sign-on by introducing a secure middle layer that authenticates

the user once and then automatically detects and handles subsequent requests for user credentials. Specifically, it uses patented client-side intelligence to respond to requests for user credentials (username/ID, password, and so on) from any Windows, Web, or Mainframe/Host application. Tivoli Access Manager for Enterprise Single Sign-On supports authentication from any authenticator (for example, Passwords, Biometrics, Tokens/Smart Cards) and authentication service (for example, Windows, Entrust PKI, RSA Keon PKI, LDAP directory).

The benefits of desktop SSO are depicted in Figure 4-3.

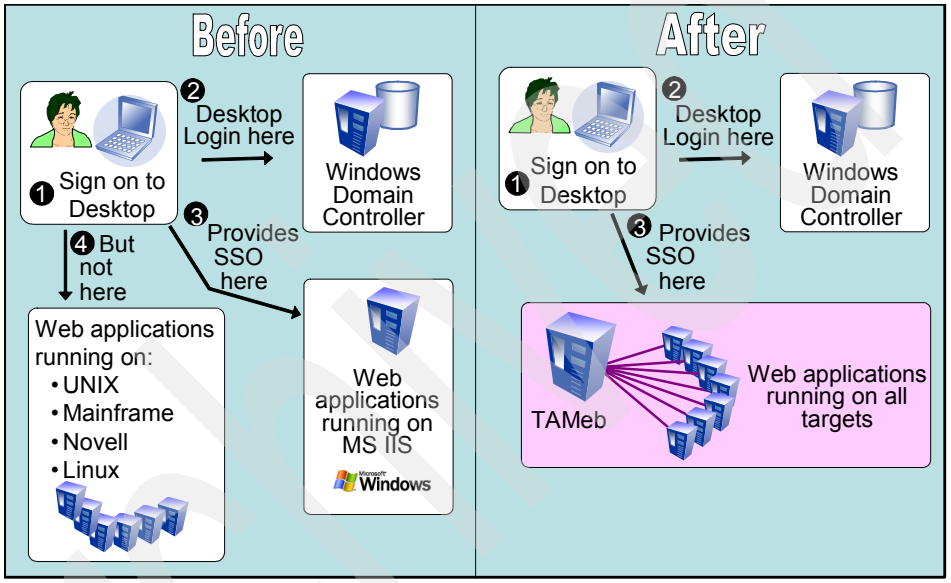


Figure 4-3 Benefits of desktop SSO

The next section discusses how Web SSO plays a crucial role in an enterprise SSO solution.

4.4 Web single sign-on

The predominant computing model today is the Web model, involving HTTP/HTTPS transactions with applications on Web servers, application servers, or both. Many legacy client-server applications are being converted over to the Web model and virtually all new applications are Web-based. With the rapid introduction of new Web applications, each requiring user authentication, companies must adopt a login management strategy or risk overwhelming their user population and consequently reducing security.

Tivoli Access Manager for e-business takes a reverse Web proxy approach to solving this problem. The reverse proxy is called WebSEAL and it intercepts Web traffic destined for the corporate Web applications as shown in Figure 4-4.

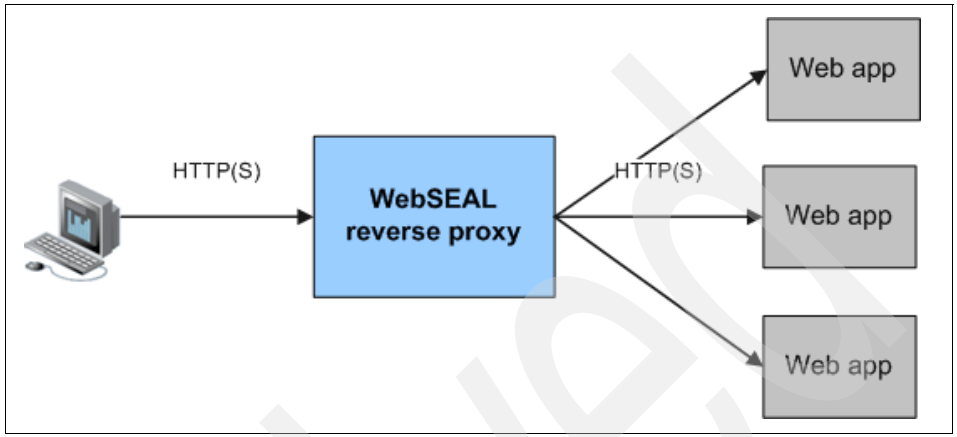


Figure 4-4 Tivoli Access Manager for e-business Web SSO solution

Before users can access the desired Web application, they must first authenticate to the WebSEAL reverse proxy. After they are authenticated, WebSEAL checks its policy database to verify whether the user is allowed to access the requested resource. If granted, the user is presented with the originally requested back-end resource without any further authentication.

When a Web application is secured through WebSEAL, it should be configured to allow Web traffic only from the WebSEAL server. With this network security in place, legacy applications may disable authentication and remove the original user accounts from the Web servers. This means that users no longer have to remember the login for that Web application and administrators do not have to manage those accounts anymore.

When all corporate Web applications are brought behind the WebSEAL server, users will only have to remember one login for all corporate Web applications.

To facilitate browser/Web server interactions, Tivoli Access Manager for e-business supports the following:

- Web trust configurations—using IBM WebSphere Application Server SSO capabilities and others
- Basic authentication SSO

- Forms-based SSO
- Lightweight third-party authentication (LTPA) SSO
- Passing user information in the HTTP header

Because customers have used Tivoli Access Manager for e-business and its precursors to solve Web SSO issues since the early 1990s, there have been many additions to its Web SSO capabilities, addressing a wide variety of business needs. Consequently, Tivoli Access Manager for e-business can be used to address desktop SSO, back-end and portal SSO, three-tier SSO, SSO to host application emulators, and federated SSO. Only a robust Web SSO solution addresses all of these areas.

4.4.1 Desktop SSO

For companies not using Tivoli Access Manager for Enterprise Single Sign-On, Tivoli Access Manager for e-business can be used to integrate desktop sign-on with Web sign-on. A user logging onto Windows is automatically logged onto Tivoli Access Manager for e-business and consequently has access to all secured Web applications without further need to sign on. This is sometimes called *Kerberizing* Tivoli Access Manager for e-business because the technology is based on the Kerberos protocol that Microsoft uses in its Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) and Microsoft Windows NT® LAN Manager (NTLM) implementations.

4.4.2 Back-end and portal SSO

It is not uncommon for companies to implement a so-called SSO solution for a portal only to find that they still get many password prompts. This is because inferior SSO solutions handle the link between the Web browser and the portal but not those between the portal and its portlets, which connect to other applications that need ID and password combinations. But with Tivoli Access Manager for e-business, user information can be passed to an application server or portal server, and that information can be used to build a credential appropriate to the back-end application environment.

To extend SSO to back-end applications and portals, Tivoli Access Manager for e-business includes the following:

- Java Authentication and Authorization Services (JAAS) standardized support for programmatic security.
- J2EE standardized support for declarative security.

- A technology preview that enables programmatic and declarative security for .NET applications.
- Special support integrated with the WebSphere Portal credential vault to extend SSO support to the portal's back-end applications.

4.4.3 Three-tier SSO

Mainframe applications protected by IBM RACF are widely appreciated for their high degree of security. Many businesses have Web-enabled these applications to extend their value, but not every SSO solution can manage authentication with mainframe applications. Tivoli Access Manager for e-business works in concert with WebSphere software, RACF, and J2EE Connector Architecture (JCA) capabilities to map user information for use in each environment that is involved in a user's request for data as shown in Figure 4-5.

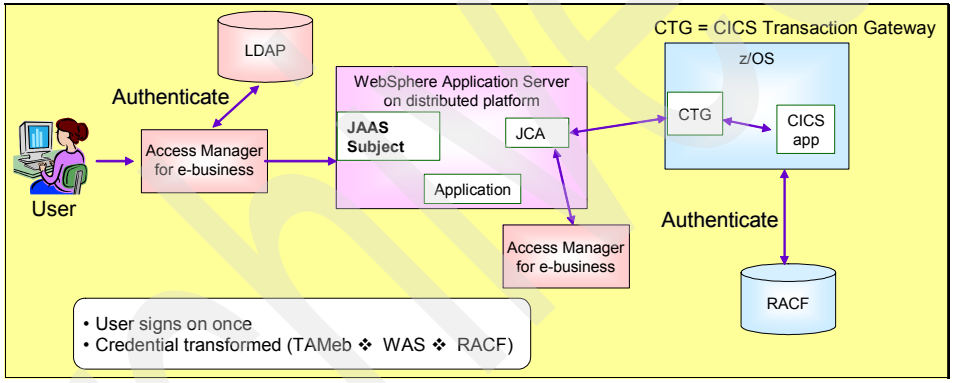


Figure 4-5 Three-tier SSO

Because such transactions involve browsers, middle-tier servers and enterprise servers, they are typically called *three-tier transactions*.

4.4.4 SSO to host application emulators

Another set of applications that have had their value extended by Web enablement are emulation applications running on zSeries, iSeries, and DEC/UNIX. The integration of Tivoli Access Manager for e-business with IBM WebSphere Host Access Transformation Services and IBM WebSphere Host On-Demand enables clients to provide SSO to these emulation applications.

4.5 Federated single sign-on

Many businesses are moving toward federated configurations to cost-effectively introduce partner-hosted capabilities into their customers' Web experiences. These environments typically involve a business that has partner relationships, where the partner is not necessarily using the same software as the business itself. Consequently, it is essential that federated software supports the latest interoperability standards used in SOA-based environments: SAML, Liberty Alliance, and WS-Federation.

Federated single sign-on protocols like SAML define methods for securely transferring a user's identity between security domains over the Internet. This typically involves a pair of companies who have formed a business relationship in which one company is consuming a service from the other. This transferring of the identity means that a service provider need not manage passwords and the user can access external services without having to authenticate again. This is what is known as *federated single sign-on*.

The powerful IBM solution for addressing federated SSO is Tivoli Federated Identity Manager, which includes Tivoli Access Manager for e-business. Together, these technologies provide robust management of identities involved in business-to-business SSO transactions. A key aspect of Tivoli Federated Identity Manager is its support of three key federated SSO interoperability standards: SAML, Liberty Alliance, and WS-Federation. This is important because in business-to-business exchanges you cannot always be sure which protocol your partner can support.

For companies, who want to test the waters of federated SSO, but do not want to invest in a full-blown enterprise solution, the Tivoli Federated Identity Manager Business Gateway (see Figure 4-2 on page 153) implements the SAML SSO protocol and provides a push button installation. It can be used by small and large companies alike to get quickly up and running with a business partner.

Customers looking to leverage federated configurations to expand their business with relatively minor investments can now do so with great security, thanks to the combination of Tivoli Federated Identity Manager and Tivoli Access Manager for e-business.

4.6 Enjoy security management benefits beyond SSO

Tivoli Access Manager for e-business not only delivers substantial SSO value, it also provides a number of additional security management benefits, including:

- Authorization for Web applications, enabling uniform application of policies that specify who can and who cannot access sets of resources.
- Reverse proxy, protecting intranet, Web and application servers from Internet access (and, optionally, from intranet access).
- Front-end authentication for applications:
  - Out-of-the-box support for multiple authentication mechanisms (including user identities and passwords, certificates and tokens), without requiring modification of back-end applications to support these technologies.
  - Switch user capability (where an administrator can take over a user's session), and authentication step-up and forced reauthentication (for accessing highly sensitive target data and applications), essential authentication options for some businesses.
- Audit capabilities, when combined with the clear, unified access-control policy, can be a key enabler of audit readiness and compliance with such regulations as Sarbanes-Oxley. Tivoli Access Manager for e-business is designed to help companies maintain and certify the validity of their records and disclosures of pertinent information.

In addition to its federated SSO capabilities, Tivoli Federated Identity Manager extends the Web services security function of WebSphere and WebSphere Web Services Gateway by:

- Expanding support for security token types, which allows out-of-the-box use of SAML and Liberty tokens.
- Mapping user identities received from another domain to identities understood locally, and then mapping and adding attributes as necessary.
- Authorizing local identities for access to requested Web services, ensuring only legitimate use of the Web services.

4.7 Conclusion

In this chapter, we saw that by dividing the SSO problem into three separate classes (desktop, Web, and federated), IBM has been able to provide SSO technologies that successfully address each area.

Tivoli Access Manager for e-business delivers SSO in the area where its need is most prevalent today—the Internet. Additionally, the software works with Tivoli

Federated Identity Manager to address federated and Web services SSO. Finally, Tivoli Access Manager for Enterprise Single Sign-On addresses existing legacy client/server configurations to close the loop on the single sign-on problem.

Part 2

Managing access control

In Part 2, we discuss the Tivoli solutions that address the access control domain of the overall security architecture. Access control information, which generally evolves around authentication and authorization mechanisms, is handled mainly by IBM Tivoli Access Manager and its resource managers. Access Manager handles a multitude of integration aspects with all sorts of IT infrastructures and application environments, which are detailed throughout this part of the book.

5

Access Manager core components

In this chapter we introduce the family of access control products offered by Tivoli and how they relate to each other. The focus of the chapter, however, is to introduce the core components of IBM Tivoli Access Manager.

The following products make up the Access Manager family:

- Access Manager for e-business
- Access Manager for Business Integration
- Access Manager for Operating Systems

Note: Although Tivoli Access Manager for Enterprise Single Sign-On bears the same naming of the components mentioned above, it does not share the same core components. The components for Access Manager for Enterprise Single Sign-On are discussed in Chapter 15, “Access Manager for Enterprise Single Sign-On” on page 449.

The components that make up Access Manager are discussed to provide the foundation for introducing the elements of the Access Manager architecture. There are three types of Access Manager components:

- Base components, which are generally common to all Access Manager installations.

- Resource managers, which support authorization for specific application classes.
- Interface components, which permit application programs to directly interact with Access Manager functions.

Before we start addressing these major components, we introduce the IBM Tivoli Access Manager family.

5.1 Tivoli Access Manager family

IBM Tivoli Access Manager (Tivoli Access Manager) is an authentication and authorization solution for corporate Web, client/server, and existing applications. Tivoli Access Manager enables you to control user access to protected information and resources. By providing a centralized, flexible, and scalable access control solution, Tivoli Access Manager enables you to build secure and easily managed network-based applications and e-business infrastructure. Tivoli Access Manager supports authentication, authorization, audit and logging, data security, and resource management capabilities.

Tivoli Access Manager provides:

- Authentication framework

  Tivoli Access Manager provides a wide range of built-in authenticators and supports external authenticators. The wide range of available authentication mechanisms are discussed in Chapter 9, “Authentication and single sign-on with Access Manager for e-business” on page 279.

- Authorization framework

  The Tivoli Access Manager authorization service, accessed through a standard authorization application programming interface (authorization API), provides permit and deny decisions on access requests for native Tivoli Access Manager servers and other applications.

  The authorization service, together with resource managers, provides a standard authorization mechanism for business network systems.

  Tivoli Access Manager can be integrated into existing and emerging infrastructures to provide secure, centralized policy management capability.

  More about the authorization framework is discussed in Chapter 10, “Access Manager authorization” on page 323.

Some existing Access Manager resource managers include:

- IBM Tivoli Access Manager for e-business

  As part of Access Manager for e-business, WebSEAL manages and protects Web-based information and resources.

- IBM Tivoli Access Manager for Operating Systems

  Access Manager for Operating Systems provides a layer of authorization policy enforcement on Linux and UNIX systems in addition to that provided by the native operating system. Existing applications can take advantage of the Tivoli Access Manager authorization service as well as provide a common security policy for the entire enterprise. Refer to Chapter 12, "Access Manager for Operating Systems" on page 381 for more information.

- IBM Tivoli Access Manager for Business Integration

  Access Manager for Business Integration provides a security solution for IBM MQSeries® and IBM WebSphere MQ messages and is discussed further in Chapter 14, "Access Manager for Business Integration" on page 425.

Note: Access Manager for Operating Systems and Access Manager for Business Integration are sold as separate products.

5.1.1 Access Manager for e-business

Tivoli Access Manager for e-business provides robust, policy-based security to a corporate Web environment. This means several things. Authentication of users, control of access privileges, auditing, single sign-on, high availability, and logging are all essential elements of any security management solution. The control of access privileges is expansive, with WebSEAL or the Plug-in for Web servers component able to manage access control to Web servers. For application integration, Access Manager for e-business provides several plug-ins such as Microsoft .NET and BEA WebLogic. These provide advanced capabilities to manage access control at the application level.

5.1.2 Access Manager for Operating Systems

Tivoli Access Manager for Operating Systems provides a layer of authorization policy enforcement in addition to that provided by the native operating system for Linux and UNIX-based systems. An administrator defines additional authorization policies by applying fine-grained access controls that restrict or permit access to key system resources. Controls are based on user identity, group membership, the type of operation, time of the day or day of the week, and the accessing application. An administrator can control access to specific file resources, login and network services, and changes of identity. These controls

can also be used to manage the execution of administrative procedures and to limit administrative capabilities on a per-user basis.

In the context of legal and regulatory compliance, strong mechanisms are provided to audit authorization decisions as well as granted and denied access to system resources.

5.1.3 Access Manager for Business Integration

Tivoli Access Manager for Business Integration operates in conjunction with IBM Tivoli Access Manager for e-business. Together, these software applications provide a security solution for IBM WebSphere MQ products.

With Tivoli Access Manager for Business Integration you can:

- Secure sensitive or high-value messages processed by IBM WebSphere MQ.
- Control which users have access to specific queues.
- Detect and remove rogue or unauthorized messages before they are processed by a receiving application.
- Generate detailed audit records showing which messages were expressly authorized and encrypted.
- Define authorization and data protection policies centrally for IBM WebSphere MQ resources (getting and putting messages to queues) using a Web browser or command line.
- Provide integrity and privacy protection for your data as it flows across the network and while it is in a queue.
- Secure existing off-the-shelf and customer-written applications for IBM WebSphere MQ.

5.2 Architectural perspective

To illustrate the value of the Access Manager solution, we describe the product in terms of a greater enterprise architecture. Throughout this book references from 2.1, “Common security architecture subsystems” on page 20 are used to place the Tivoli Security products within an overall architecture perspective.

5.2.1 Design principles

The design of any architecture must be based on clearly defined and articulated principles that form a foundation for the design process. That is, the principles describe the objectives of the solution. Whenever in doubt about a design

decision, the principles should be used to map a path forward and to justify the overall design.

Some key principles can be applied to an access control solution:

- The security solution must have a central point of authority for security-related information. This authority must support both centralized and distributed management.
  - Motivation: This principle drives the need for one source of authoritative, security-related policy within an organization. It enables a consistent policy to be applied across applications, systems, and throughout the organization while providing a flexible administration framework that fits into and enhances an organization's operation capabilities.
  - Implication: This principle implies a high degree of integration, broad coverage, and flexibility required from the products that are chosen to support it. Integration is one of the greatest challenges.
- Access decisions must be evaluated where and when they are required, not at the beginning of a transaction. Gated controls should be employed throughout the solution. Putting all controls at the front door puts too much emphasis on the concept of trust (that is, I have let you into my house and now you can do whatever you like), creating an inherently less secure system.
  - Motivation: The drivers for this principle are increased security and performance:
    - Increased security through more checks of a user's or transaction's authority to perform a function.
    - Increased performance as decisions get made when a user requires something, meaning that unnecessary decisions about a user's potential activity will not be made up front.
  - Implication: Requires good integration capability to enable a common security service to permeate an environment. The majority of applications must be able to use the security services.
- Sufficient logging is required to capture all authentication and access control decision events and logs. The level of logging should be based on business and security requirements, hence the security solution should provide comprehensive and flexible logging coverage, allowing it to be customized.
  - Motivation: Because no security solution is foolproof, it is essential to keep good records of the transactions performed by the security system. An easily manageable method of dealing with these records is essential.
  - Implications: Strong integration is required to provide logging across multiple systems. Mechanisms must be in place to collect, filter, analyze, and report on audit data.

These principle are not intended to be comprehensive, but to highlight some core objectives of the security solution.

Tip: When defining design principles it is important to specify the motivations and implications of each principle. This gives background as to why the principle was accepted and developed and, more important, it describes the consequences of adopting a particular principle.

One of the core implications of the principles just listed is that integration throughout the security solution will always be a huge issue within an enterprise context. Access Manager offers virtually full security coverage when it comes to access control for Web-based applications, addressing both the depth (Access Manager for Operating Systems) and breadth (Access Manager for Business Integration) of enterprise access control security solutions.

The Access Manager family supports all of these principles. The Access Manager family of products, when integrated throughout an environment, provides a comprehensive access control capability. The breadth of the Access Manager solution, along with its open architecture and interfaces, means that it is a perfect solution to provide the majority of an enterprise's access control capabilities.

Access Manager provides the core security functions for Web-based enterprise solutions. Integrated with Tivoli Identity Manager, Tivoli Directory Integrator, and Tivoli Security Operations Manager, the Tivoli security products provide the access control, identity management, and threat management capabilities required for any enterprise.

5.2.2 Security subsystems

As discussed in 2.1, "Common security architecture subsystems" on page 20 there are common subsystems used in a security solution. Access Manager is primarily an access control solution, addressing two of the common subsystems:

- Access control: Access Manager is used to authenticate users and to enforce security policy at an application and system level.
- Auditing: The Access Manager components and infrastructure provide a comprehensive logging framework that can be integrated with any threat management system.

Access Manager utilizes all the subsystems, but these two are fundamental to the position of Access Manager within an overall Enterprise Architecture.

From an Enterprise Architecture perspective, other products will be required to satisfy other solution requirements. One product cannot be everything to

everyone. However, the Access Manager family provides a comprehensive, integrated security solution that is powerful in its coverage, scalability, and reliability.

5.2.3 Access control subsystem

An access control subsystem is responsible for data and component protection by providing mechanisms for identification and authentication as well as authorizing component access. In addition to these major functions, it also provides security management and cryptographic support.

Figure 5-1 on page 170 shows a use case model of an access control subsystem. The physical view shows the systems involved in the transaction. The component view depicts the information flow control function that examines messages being sent and, based on a set of rules, will allow valid messages to flow. Invalid messages are rejected and recorded. The logical view breaks down the access control process into distinct functions.

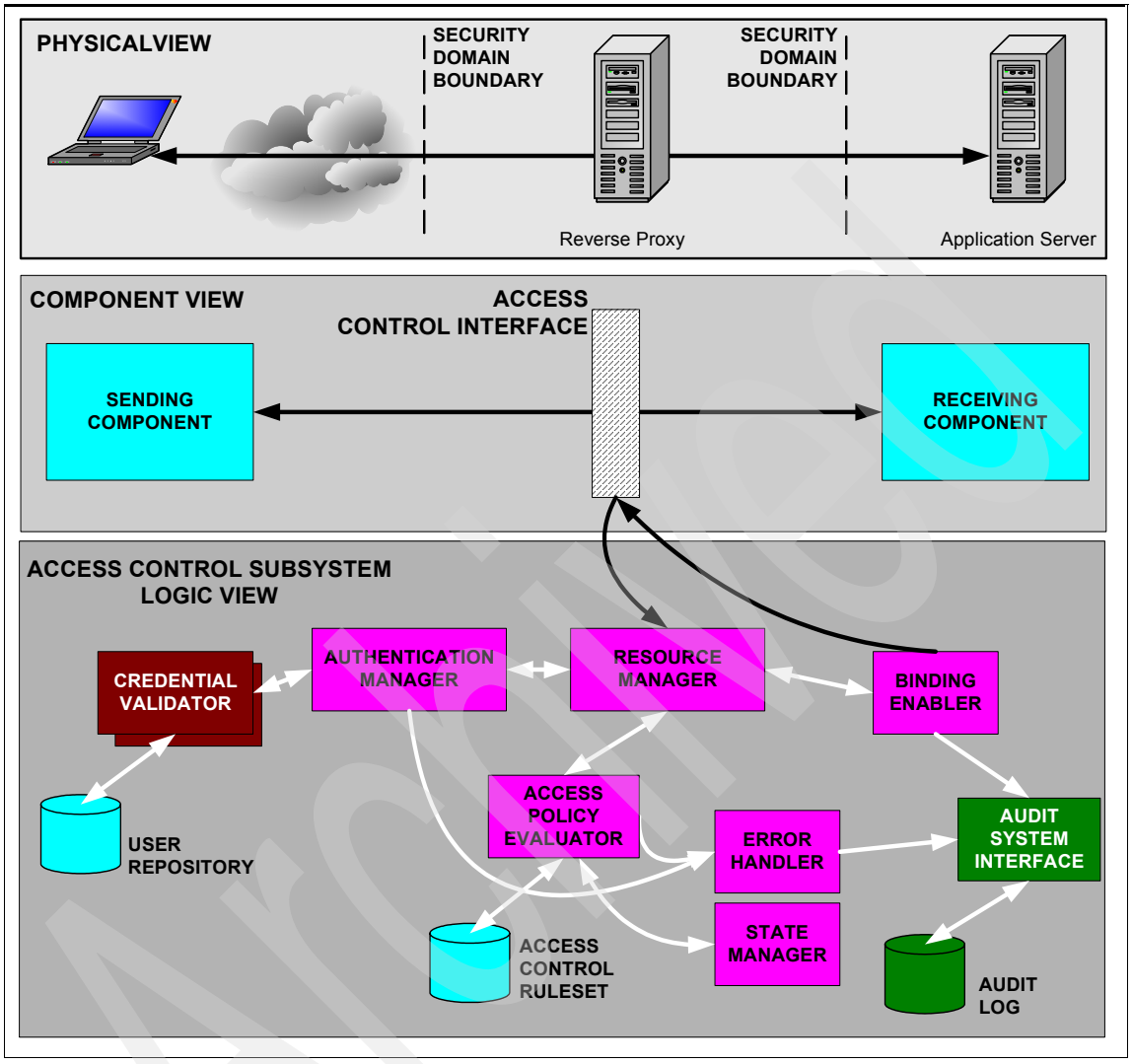


Figure 5-1 Access control subsystem

The *Resource Manager* is the major component involved in an access control decision. It is positioned between two security domain boundaries, so every transaction or information request has to be routed through this component.

If the sending component has not yet been authenticated, the Resource Manager involves the *Authentication Manager* and *Credential Validator* service in order to verify the requester and issue a credential package that will be returned to the Resource Manager. If the requester could not be authenticated successfully, the

Error Handler will be involved, and the *Audit System Interface* writes an entry into the *Audit Log*.

If the sending component has already been authenticated, the Resource Manager sends the authorization requests to the *Access Policy Evaluator*, which first uses the *State Manager* to verify the current status of the session.

If the session is still active and everything proves valid, the Access Policy Evaluator proceeds with the evaluation of the request by applying access control rules from the *Access Control Ruleset* database.

If access is granted, the Access Policy Evaluator updates the information in the State Manager and hands the task over to the *Binding Enabler*. If configured, the Binding Enabler might ask the Audit System Interface to write a positive log entry.

If access is not granted, the Access Policy Evaluator updates the information in the State Manager and hands the task over to the Error Handler, which writes a log entry. It then informs the Binding Enabler of the negative decision, which in return informs the requester of the denied access.

This example use case flow demonstrates what sort of components are required within an Access Control system and how they relate to each other. Use cases are generally used to describe real solution component interactions and form a very valuable tool when determining the best possible design.

In the following sections and chapters, the Access Manager family of products is described from an architectural perspective. The functional components that make up the products are described and real world examples are used to illustrate the products applications.

5.3 Base components

Access Manager provides several components that support basic product functionality. The Access Manager base consists of a small set of architectural core components and management facilities that generally are required to support and administer the environment. The components are common across the Access Manager family of products.

5.3.1 Overview

Access Manager's base functions are provided through a set of core components and various management components.

Underlying components

Access Manager is based on two components:

- A user registry
- An Authorization Service consisting of an authorization database and an authorization engine

These components support the core functionality that must exist for Access Manager to perform its fundamental operations, which are:

- Knowing the identity of who is performing a particular operation (users)
- Knowing the roles associated with a particular identity (groups)
- Knowing what application entities a particular identity may access (objects)
- Knowing the authorization rules associated with application objects (policies)
- Using this information to make access decisions on behalf of applications (authorization)
- Auditing and logging all activity related to authentication and authorization

In summary, a user registry and an Authorization Service are the fundamental building blocks upon which Access Manager builds to provide its security capabilities. All other Access Manager services and components are built on this base.

Management components

The Access Manager environment requires certain basic capabilities for administrative control of its functions. Management facilities are provided through the following base components:

- The Policy Server, which supports the management of the authorization database and its distribution to Authorization Services.
- A Policy Proxy Server, which provides a mechanism for resource managers to access Policy Server functionality without a direct connection to the master Policy Server.
- The pdadmin utility, which provides a command line capability for performing administrative functions such as adding users or groups.
- The Web Portal Manager, which provides a browser-based capability for performing most of the same functions provided by the pdadmin utility.

- The administration API, on which the pdadmin utility and the Web Portal Manager are built, enables performance of program initiated level administration tasks and queries.

5.3.2 User registry

Access Manager requires a user registry to support the operation of its authorization functions. Specifically, it provides:

- A database of the user identities that are known to Access Manager
- A representation of groups in Access Manager that may be associated with users
- A data store of other metadata required to support authorization functions

Identity mapping

While it can be used in authenticating users, this is not the primary purpose of the user registry. An application can authenticate a user via any mechanism it chooses (ID/password, certificate, and so on), and then map the authenticated identity to one defined in the user registry. For example, consider a user John who authenticates himself to an application using a certificate. The application then maps the DN in John's certificate to the Access Manager user named john123. When making subsequent authorization decisions, the internal Access Manager user is john123, and this identity is passed between the application and other components using various mechanisms, including a special credential known as a *Privilege Attribute Certificate* (PAC).

Note: One of the primary goals of the authentication process is to acquire credential information describing the client user. The user credential is one of the key requirements for participating in the secure domain.

Access Manager distinguishes the authentication of the user from the acquisition of credentials. A user's identity is always constant. However, credentials, which define the groups or roles in which a user participates, are variable. Context-specific credentials can change over time. For example, when a person is promoted, credentials must reflect the new responsibility level.

The authentication process results in method-specific user identity information. This information is checked against user account information that resides in the Access Manager user registry. Access Manager maps the user name and group information to a common domain-wide representation and format known as the Privilege Attribute Certificate (PAC).

Method-specific identity information, such as passwords, tokens, and certificates, represent physical identity properties of the user. This information can be used to establish a secure session with the server.

The resulting credential, which represents a user's privileges in the secure domain, describes the user in a specific context and is valid only for the lifetime of that session.

Access Manager credentials contain the user identity and groups where this user has membership.

User registry structure

The user registry contains three types of objects:

- User objects, which contain basic user attributes.
- Group objects, which represent roles that may be associated with user objects.
- Access Manager metadata objects, which contain special Access Manager attributes that are associated with user and group objects. The metadata includes information that helps link an Access Manager user ID to its corresponding user object.

Tivoli Access Manager has altered the way it stores a user's metadata objects in the directory. It has migrated to a minimal model that minimizes the disruption to an existing DIT structure. All data for Tivoli Access Manager can now be stored under a separate secAuthority=Default suffix leaving any existing suffixes to coexist in the directory. Figure 5-2 on page 175 illustrates how Tivoli Access

Manager data can be added to a directory already containing user and group information without impacting the pre-existing suffix.

Note: The *minimal data model* is optional and is used for new Access Manager configurations by default. The data model used by previous versions of Access Manager (called the *standard data model*) is still supported and should be chosen if there will be any previous versions of Access Manager components or resource managers in the enterprise, since previous versions will not recognize or support the minimal model. For existing Access Manager configurations, which upgrade to version 6.0, the current standard data model will remain and can be used unchanged, no migration is required.

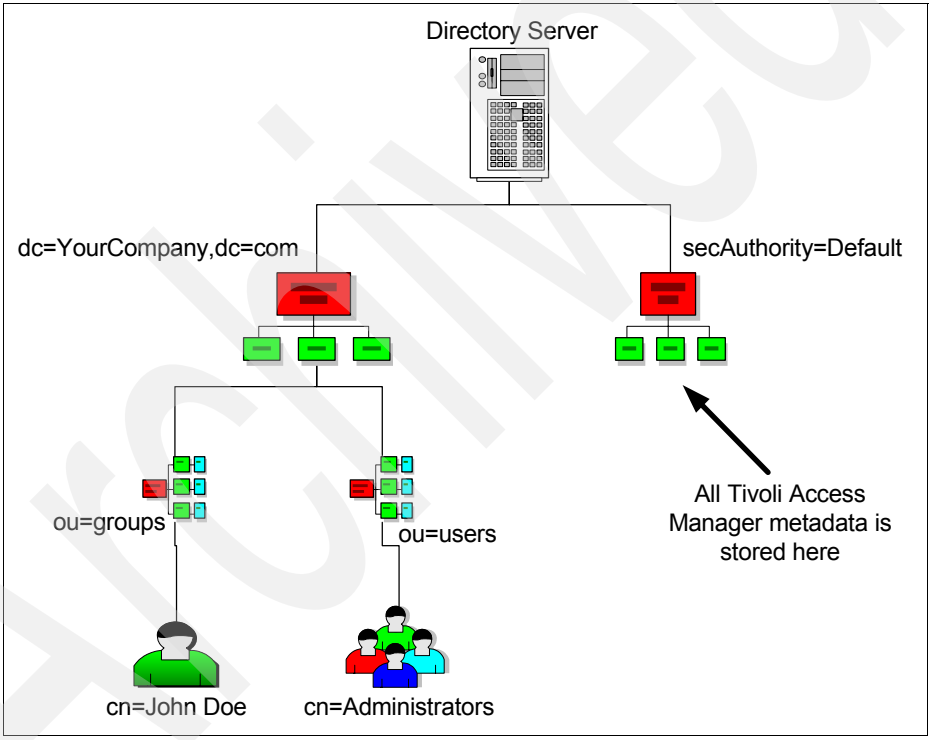


Figure 5-2 Tivoli Access Manager LDAP directory storage of metadata

The default user registry is LDAP-based, and Access Manager consolidates its registry support around a number of LDAP directory products.

Access Manager can use the following directory products for its user registry:

- IBM Tivoli Directory Server
- Novell eDirectory
- Sun ONE Directory Server
- Sun Java System Directory Server
- Microsoft Active Directory
- Lotus Domino Server
- IBM z/OS Security Server LDAP Server

The IBM Tivoli Directory Server is included with Access Manager and is the default LDAP directory for implementing the user registry. For the latest list of supported user registries refer to the *IBM Tivoli Access Manager for e-business Version 6.0 Release Notes,* SC32-1702.

Access Manager components support the use of directory replicas, peer-to-peer (multi-master) replication, and directory partitioning. It is recommended that a directory architecture be completed to ensure the directory environment will perform as expected with Tivoli Access Manager and any other applications that may want to participate in directory services. Minimal recommendations for the directory in regards to Tivoli Access Manager are a master-replica topology. For a more detailed discussion on directory technology refer to Chapter 3, “Directory technologies” on page 49.

Directory schema

To support its critical and private registry data, Access Manager requires certain support in the directory schema. Certain object classes and attributes are specific to Access Manager and are configured as needed during product installation. Access Manager, however, only adds new subclasses to existing directory objects (for example, inetOrgPerson).

Attention: While it might seem relevant to inquire about the details of the directory schema that Access Manager uses, such information is not necessarily useful (and in fact may be undesirable to have). It is important to keep in mind that Access Manager components are the exclusive users of these special object classes and attributes. The schema definitions and their usage can change from release to release. As such, application components should not assume any knowledge of Access Manager-specific schema definitions or how they are used. Instead, application interaction with registry information or functions should only be performed using published Access Manager interfaces.

5.3.3 Authorization database

Separate from the user registry, Access Manager uses for its authorization functions a special database containing a virtual representation of resources it protects. Called the *protected object space*, it uses a proprietary format and contains object definitions that may represent logical or actual physical resources. Objects for different application types may be contained in different sections of the object space, and the object space may be extended to support new application types as required.

The security policy for these resources is implemented by applying appropriate security mechanisms to the objects requiring protection. Security mechanisms are also defined in the authorization database, and include:

- Access control list (ACL) policy templates

  ACLs are special Access Manager objects that define policies identifying user types that can be considered for access, and specify permitted operations. In the Access Manager model, ACLs are defined separately from and then attached to one or more protected objects. So an ACL has no effect on authorization until it becomes associated with a protected object.

  Access Manager uses an inheritance model in which an ACL attached to a protected object applies to all other objects below it in the tree until another ACL is encountered.

- Protected object policy (POP) templates

  A POP specifies additional conditions governing the access to the protected object, such as privacy, integrity, auditing, and time-of-day access.

  POPs are attached to protected objects in the same manner as ACLs.

- Extended attributes

  Extended attributes are additional values placed on an object, ACL, or POP that can be read and interpreted by third-party applications (such as an external Authorization Service).

- Authorization rules (Rules)

  Authorization rules are defined in XSL (eXtensible Stylesheet Language) to specify further conditions that must be met before access to a resource is permitted. Rules enable you to make authorization decisions based on the context and the request environment, as well as who is attempting the access and what type of action is being attempted. These conditions are evaluated as a Boolean expression to determine whether the request should be allowed or denied.

Figure 5-3 depicts the relationships between the protected object space, ACLs, POPs, and Rules. The different security mechanisms are discussed in detail in Chapter 10, “Access Manager authorization” on page 323.

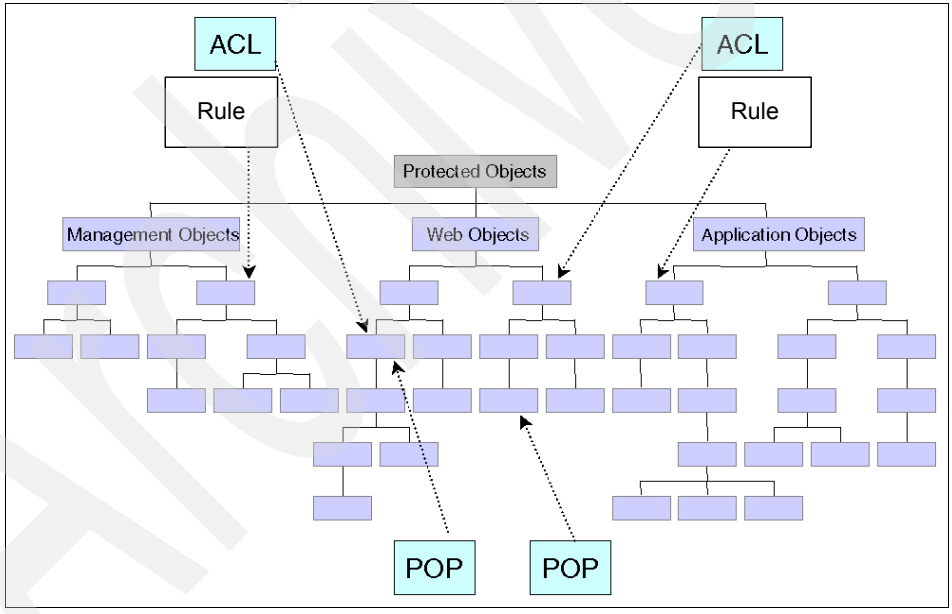


Figure 5-3 Relationship between the protected object space, ACLs, POPs, and Rules

Successful implementation of a security policy requires that the different content types are logically organized (and that the appropriate ACL, POP, and Rule policies are applied). Access control management is simplified by structuring the protected resources in such a way as to minimize the number of ACL, POP, and

Rule attachments required to implement the security policy, and thus gaining maximum benefit from the sparse ACL model that we implement.

5.3.4 Policy Server

The Access Manager Policy Server maintains the master authorization database for the secure domain. This server is key to the processing of access control, authentication, and authorization requests. It also is responsible for distributing and updating all authorization database replicas and maintaining location information about other Access Manager servers in the secure domain.

Multi-Domain Policy Server

There can only be a single Policy Server in an Access Manager domain. There can, however, be multiple secure domains contained within a single Policy Server. Each domain has its own authorization database, resource managers, administrative users and groups, and Global Sign-On (GSO) information. In addition, domains can either share users and groups or each have their own set of users and groups. Management tools may also be shared between domains or allocated on a per domain basis. Figure 5-4 illustrates the relationship between Access Manager components in a multi-domain environment.

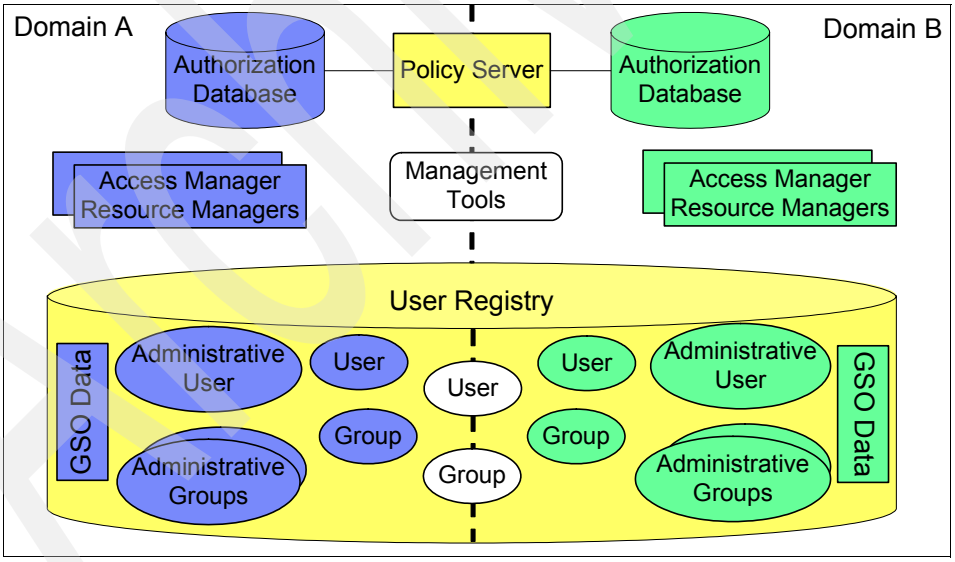


Figure 5-4 Access Manager components in a multi-domain environment

In a single domain environment, the default domain is the only domain used. In a multi-domain environment, the default domain becomes the management

domain. The Policy Server will always belong to this domain. All domains are created and deleted from the management domain.

Figure 5-5 illustrates the relationship between the Policy Server, multiple domains and their corresponding authorization databases.

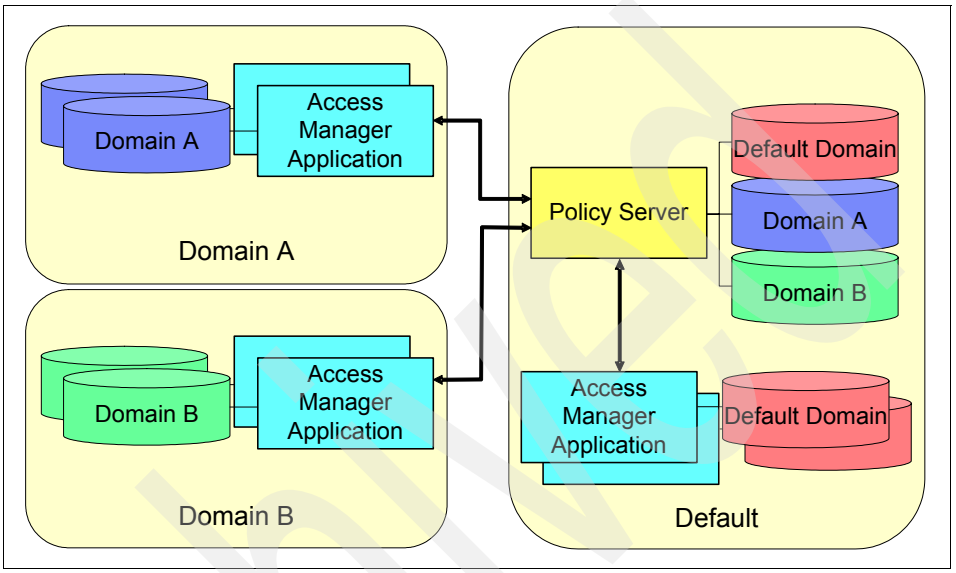


Figure 5-5 Multiple domains with multiple authorization databases

There are many valid reasons why an enterprise might consider the multiple domain model when developing their security architecture. One of the main reasons is the need to segment security completely while still sharing the same user base. When using multiple domains, completely separate policies can be set up for each domain. There is no possibility that a security policy from one domain can conflict with a security policy of another. Also, since the administrative functionality is completely separated, an administrator from one domain has absolutely no power in another domain. Real world examples of this would be a conglomeration of companies that want to use Access Manager and has different policies (and perhaps even laws that regulate them) that prevent them from using the same security model. Another would be a development environment on which each development organization is given their own domain to prevent conflicts during the development cycle.

Standby Policy Server

To provide the redundancy for the shared data and for the functions that are provided by the Tivoli Access Manager policy server, you can install and configure a primary policy server and a standby policy server. The standby

server takes over policy server functions in the event of a system or primary policy server failure. The standby policy server acts as the primary policy server until the original primary policy server is up and running again with the standby server back to serving as the failover server. This is further discussed in 8.2, “Availability” on page 261.

5.3.5 Policy Proxy Server

The Policy Proxy Server enables Access Manager applications and authorization servers to connect to a Policy Proxy Server rather than the Policy Server. The addition of a Policy Proxy Server enables an architecture to be created where the only incoming SSL sessions to the Policy Server come from physical Policy Proxy Servers. This facilitates increased security because a firewall protecting the Policy Server only has to allow inbound connections from the Policy Proxy Server(s) rather than from all Tivoli Access Manager applications or authorization servers. The SSL session from Access Manager applications to the Policy Proxy Server(s) is independent from the SSL session from the Policy Proxy Server to the Policy Server.

The only exception to this rule is if you are using an application that uses the administration API. Because administration API applications use the SSL protocol to communicate with the Tivoli Access Manager Policy Server you have to allow direct communication between these applications and the Policy Server.

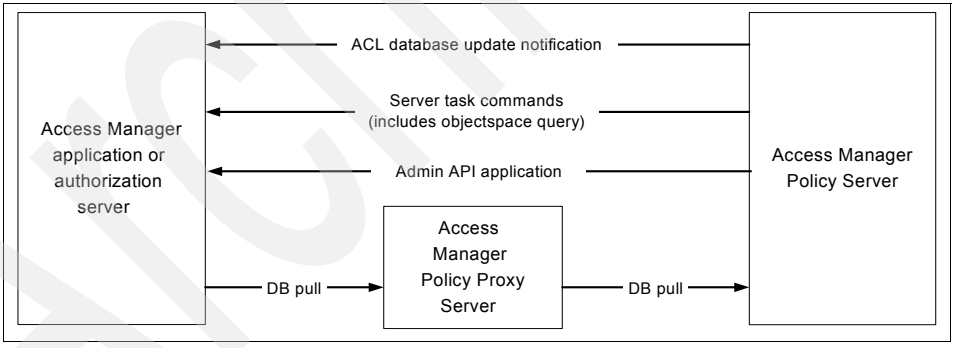


Figure 5-6 Communication flows using the Policy Proxy Server

Figure 5-6 shows the connections (and the direction of flow) between the Policy Server, a Policy Proxy Server and an Access Manager application or authorization server.

All requests inbound to the Policy Server go via the Policy Proxy Server, except for applications using the administration API. All requests outbound from the real Policy Server go directly to the Access Manager application.

ACL database caching

In addition to providing a simple proxy service, the Policy Proxy Server can also offload database replication tasks from the Policy Server by caching the ACL databases that it serves to Access Manager applications. If several Access Manager applications make requests for the same database, then the database is only transferred from the Policy Server to the Policy Proxy Server one time.

The ACL databases are cached in memory for security. There is no ACL policy database stored on the disk of the Policy Proxy Server that could be read (or modified) if the Policy Proxy Server were compromised.

The currency of the ACL database in the Policy Proxy Servers cache is checked every time a replication request is made so that there is no chance of an Access Manager application receiving an out-of-date cached version of the ACL database.

Note: The Policy Proxy Server does not perform any Policy Server functions; it simply forwards requests to the Policy Server. This means that the Policy Server is still the authoritative source for ACL database and user repository updates.

5.3.6 Authorization service

The foundation of Access Manager is its authorization service, which permits or denies access to protected objects (resources) based on the user's credentials and the access controls placed on the objects.

The Policy Server provides an authorization service that may be leveraged by applications and other Access Manager components that use the Authorization Application Programming Interface (aznAPI), described in 5.5.1, "aznAPI" on page 187. Optionally, additional Authorization Servers may be installed to offload these authorization decisions from the Policy Server and provide for higher availability of authorization functions. The Policy Server provides updates for authorization database replicas maintained on each Authorization Server.

The Access Manager authorization service can also be embedded directly within an application. In this case, the functions of an Authorization Server are contained in the application itself.

5.3.7 The pdadmin utility and administration API

pdadmin is a command-line utility that supports Access Manager administrative functions. The pdadmin utility is built on the administration API, as is the Web Portal Manager (WPM) described next. The administration API provides the core

interface into Access Manager for administrative functions. It enables the CLI and WPM interfaces and allows for program-initiated administrative functions and queries.

5.3.8 Web Portal Manager

The Access Manager Web Portal Manager provides a browser-based graphical user interface (GUI) for Access Manager administration.

A key advantage of the Web Portal Manager over the pdadmin command line utility is the fact that it is a browser-based application that can be accessed without installing any Access Manager-specific client components or requiring special network configuration to permit remote administrator access. In fact, the authorization capabilities of WebSEAL (described in 6.4.1, "WebSEAL" on page 196) can be used to control access to the Web Portal Manager. This means greater flexibility for administrators' locations with respect to the physical systems they are managing.

Administrative functionality

The Web Portal Manager was designed to be an alternative to the pdadmin command line interface (CLI) for many administrative functions. However, not all pdadmin functions are supported (such as the retrieval of server statistics) and the command line interface will still be required in certain cases. In other cases, such as exporting Access Manager authorization data, Web Portal Manager is required. Web Portal Manager also offers some key functional benefits over pdadmin such as cloning and cut/paste functionality.

Migration of data

Web Portal Manager allows for the migration of data from one Access Manager environment to another. Data is exported from the master authorization database and placed into an XML file with optional encryption. It can then be transported to a new Access Manager environment and imported.

This functionality allows for the exportation of one or more of the following items:

- Access control lists (ACLs)
- Protected Object Policies (POPs)
- Authorization Rules (Rules)
- Objects and Objectspaces including attached ACLs, POPs, and Rules

The exportation of data ensures a smooth transition from one Access Manager environment to another such as migrating from staging to production.

Delegated administration

The Web Portal Manager also provides a delegated user administration capability. This enables an Access Manager administrator to create delegated user groups and assign delegate administrators to these groups.

The initial aim of the Web Portal Manager delegate function is to enable multiple independent enterprises to manage their own user population in a single Access Manager secure domain. This functionality could be used when a service provider that uses Access Manager to provide access control to Web resources wants to allow its customers to define and manage their own user population.

Depending on their assigned roles, the delegate administrators can perform a subset of the administration functions aligning the security administration with different organization and business relationships, such as:

- Departments
- Dealerships
- Branch offices
- Partnerships
- Suppliers
- Distributors

There are four different levels of administration in Access Manager with the basic fields of action shown in Table 5-1.

Table 5-1 Delegated administration roles in Access Manager

Action/role	Domain admin	Senior admin	Admin	Support	Any other
View user	X	X	X	X	X
Reset password	X	X	X	X	
Add existing Access Manager user as an administrator	X	X	X		
Create domain user	X	X			
Remove user	X	X			
Domain control	X				

Note: Domains referenced in the above table do not correspond to Access Manager secure domains. Domains in the delegate function of Web Portal Manager are simply groups of users and functionality and have nothing to do with the separation of security policy between groups of Access Manager servers.

Architecture

The Web Portal Manager is built using Java Server Pages (JSP™), which support the various administrative functions. It uses a Web application server servlet engine; WebSphere Application Server 6.0.2 is provided with Access Manager to support this capability. Figure 5-7 provides an architectural view of how the Web Portal Manager works.

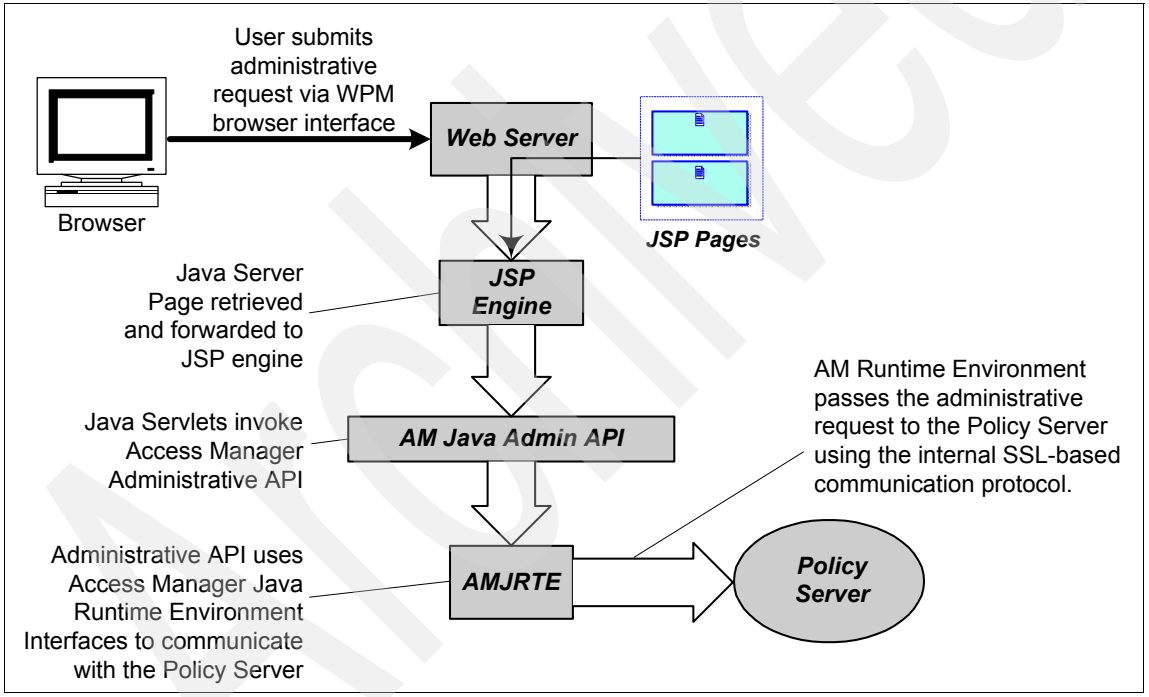


Figure 5-7 Web Portal Manager architecture

Other issues

Other issues that should be kept in mind when deploying Web Portal Manager:

- There is no limit to the number of Web Portal Manager instances that may be deployed.
- It is possible to provide access to the Web Portal Manager via a WebSEAL junction.

5.4 Resource managers

Resource managers are components that provide Access Manager authorization support for specific application types. The resource manager is responsible for the enforcement of the security policy within an Access Manager environment. The resource manager uses the *policy enforcer* to call the Tivoli Access Manager authorization service with the credentials of the user making the request, the type of access desired, and the object to be accessed. The resource manager takes the recommendation of the authorization service, performs any additional verification actions, and ultimately either denies the request or permits the request to be processed.

Figure 5-8 illustrates the interaction between the client, resource manager, authorization service, and resource.

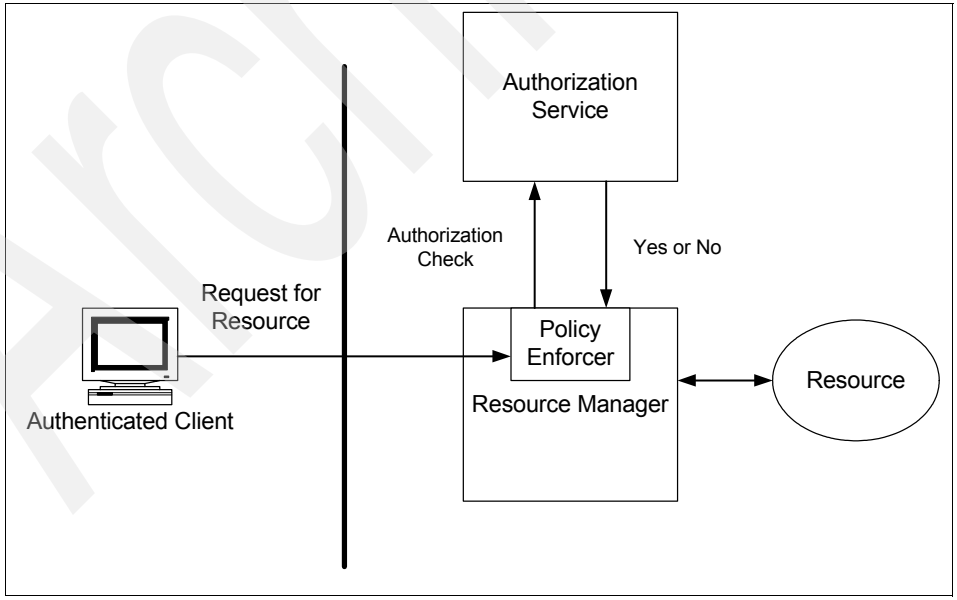


Figure 5-8 Resource Manager component interaction

5.5 Interfaces

Access Manager supports a number of programming interfaces that permit direct application interaction with its components. While these interfaces support a rich set of functionality and are useful in many situations, it is important to point out that there is substantial product function that does not require their use. Initially, many organizations do not need to utilize these interfaces, allowing rapid deployment of security components such as WebSEAL. However, as the needs of the organization evolve, these interfaces allow for a high level of security integration and customization.

5.5.1 aznAPI

The Access Manager aznAPI provides a standard programming and management model for integrating authorization requests and decisions with applications. Use of the aznAPI enables applications to utilize fined-grained access control for application-controlled resources.

Application-specific resources may be individually defined and added to the protected object space and maintained in the authorization database in the same manner that WebSEAL and other standard Access Manager blades define their respective resources. ACLs, POPs, and authorization rules may be attached to these application objects, and aznAPI calls may then be used to access the Access Manager Authorization Service to obtain authorization decisions.

In Access Manager 6.0, a new credential entitlement service has been added that allows for user policy information to be gathered from the LDAP directory and added to the user's credential. This builds upon previous functionality provided in the registry entitlement service which pulls information out of an LDAP directory and places it in the credential.

Also, Access Manager 6.0 provides two credential modification services, one for modifying attributes in a credential and another for modifying group membership within a credential.

5.5.2 Java API for Access Manager-based authorization

A powerful feature is to use Access Manager as an authentication and authorization back-end inside the Java 2 security model.

The Access Manager Authorization Java Classes provide an implementation of Java security code that is fully compliant with the Java 2 security model and the Java Authentication and Authorization Services (JAAS) extensions. More detailed information about this topic can be found in section 11.4, "Access Manager and WebSphere integration" on page 357.

5.5.3 Access Manager-based authorization for Microsoft .NET

Access Manager provides integration and support for implementing authorization for Microsoft .NET applications. Access Manager APIs are exposed at the .NET Common Language Runtime level. This exposes the functionality to all .NET languages such as Managed C++, C#, and Visual Basic® .NET. More detailed information about this topic is in section 6.4.7, “Access Manager for Microsoft .NET applications” on page 211, and Chapter 11, “Application integration” on page 347.

5.5.4 Management API

Also known as the administration API, the Management API provides C language bindings and Java admin classes to the same functions supported by the pdadmin command line utility. It may be used by custom applications to perform various Access Manager administrative functions.

5.5.5 External Authorization Service

The External Authorization Service (EAS) interface provides support for application-specific extensions to the authorization engine. This enables system designers to supplement Access Manager authorization with their own authorization models.

An EAS is accessed via an authorization “callout,” which is triggered by the presence of a particular bit in the ACL that is attached to a protected object. The callout is made directly by the Authorization Service.

In the current release of Access Manager, the EAS interface is supported via a simple Authorization Service plug-in capability. This allows an EAS to be constructed as a loadable shared library. The EAS architecture is summarized in Figure 5-9 on page 189.

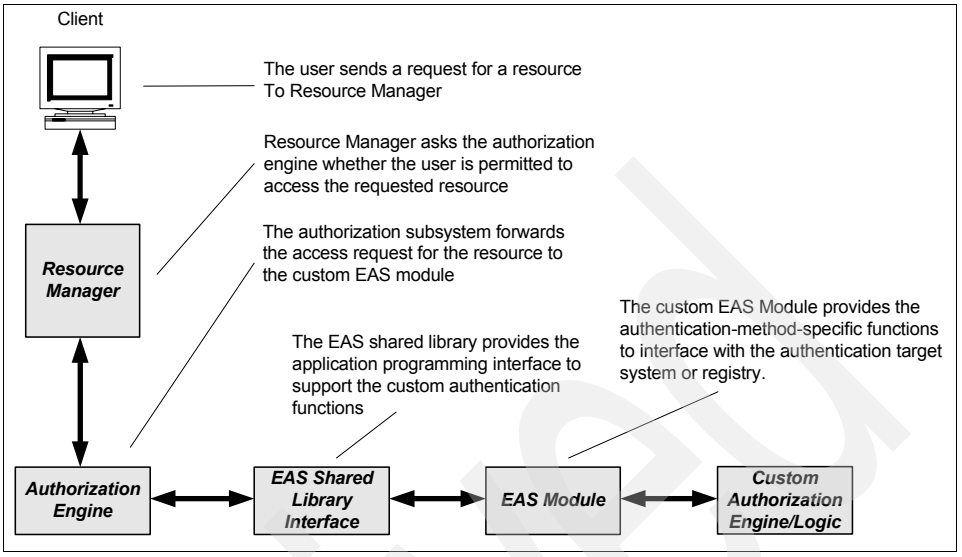


Figure 5-9 EAS architecture

After introducing the Access Manager core components we now take a closer look at Access Manager for e-business.

6

Access Manager for e-business

Web presence has become a key consideration for the majority of businesses and other organizations. Almost all organizations see the Web as an essential information delivery tool. Increasingly, however, the Web is being seen as an extension of the organization itself, directly integrated with its operating processes. As this transformation takes place, security grows in importance.

This chapter introduces the elements of the Access Manager architecture in a Web-centric environment. It describes and compares the use of the WebSEAL and Access Manager Web server plug-in resource managers and covers key architectural issues associated with any Access Manager deployment, and provides a foundation for the architectural discussions in later chapters.

6.1 Typical Internet Web server security characteristics

Perhaps the best place to begin the discussion of Access Manager architecture is with the issues typically encountered by organizations as they address Web security requirements.

It is generally accepted practice for organizations to place Internet-facing Web servers in a protected zone (also known as a demilitarized zone or DMZ), which is generally firewalled and separated from the Internet. The DMZ can provide an buffer between the external untrusted public networks of the Internet and a trusted internal corporate network. The DMZ concept enforces the *defense in depth* principle of network design, which adopts an onion skin approach. Each layer of the onion is analogous to a network zone trust level: The more sensitive the data and applications, the closer to the center of the onion they should be deployed, hence providing layers of protection from less trusted networks. Refer to 2.3.1, “Localizing a global vision” on page 31, and 2.3.2, “Network zones” on page 34.

Direct uncontrolled Internet access to such components presents a significant security exposure. For this reason, back-end components are often placed in an internal network firewalled from the Internet DMZ, leaving only the Web server component exposed to direct browser access, as illustrated in Figure 6-1. This double-firewall architecture has become common, not only for Internet application access, but increasingly for internal organization access to critical computing resources as well.

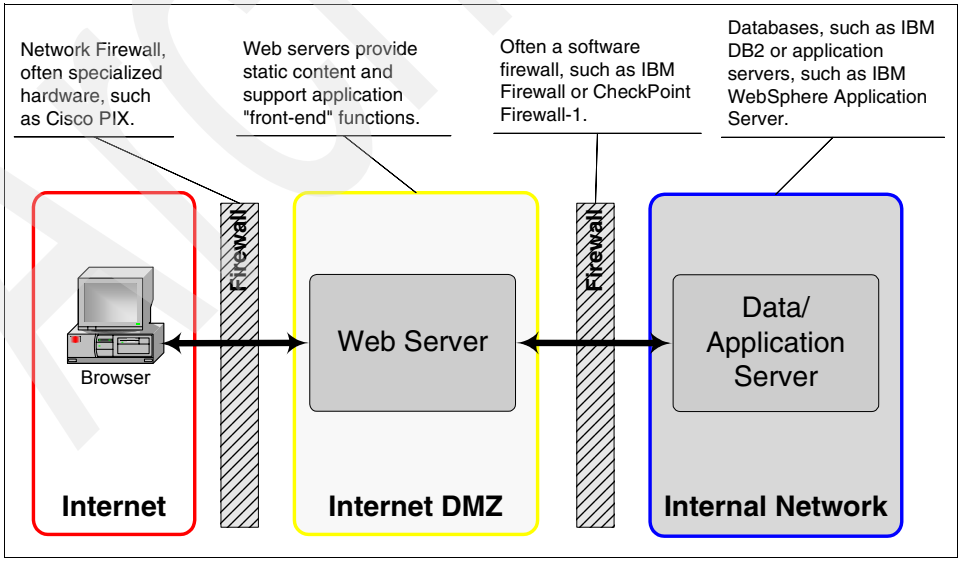


Figure 6-1 Typical advanced Web application architecture

Such architectures successfully address security from a network perspective, but they do not address a larger set of concerns, including:

- Security sensitive information may reside in the static content of Web servers (for example, Human Resources, sales, and personal information).
- Authentication/authorization may be driven by platform-specific mechanisms.
- Authentication, authorization, and audit functions may not be centralized.
- Managed security policies may be inconsistent and vary from server to server (access policies controlled by many different individuals or groups).

In such environments, there may be sensitive functions and content which, if compromised, could represent a significant business risk.

Access Manager is capable of addressing these issues. Combined with an appropriate network architecture, an organization can deploy Web content and applications with a high degree of assurance that the environment is secure and that the security functions and policies may be consistently applied.

In the following sections, we introduce the elements of Access Manager architecture, using the deployment of WebSEAL and the Tivoli Access Manager Web Plug-in as a focal point.

6.2 Web security requirement issues

The use of Tivoli Access Manager, and in particular WebSEAL and the Web Plug-in, is driven by key business requirements, which are reflected in specific design objectives, or technical requirements as outlined in the common security architecture subsytems approach discussed in 2.1, “Common security architecture subsystems” on page 20.

6.2.1 Typical business requirements

Several commonly encountered business requirements tend to drive Web security solutions such as those using WebSEAL:

- Different back-end and Web content hosting systems require users to authenticate multiple times, causing a negative user experience.

  In order to improve customer satisfaction, a method for a single user authentication has to be implemented.

- The Web-based functions of the business extend into content and applications, which increasingly require sophisticated security management.

  Almost all businesses that are on the Web encounter this. Beyond basic, static informative content, the inadequacies in simple security mechanisms typically present in many Web servers become clear. The enforcement of Web security across the enterprise cannot be successful without something more sophisticated and manageable at the enterprise level.

- Web security policies must be consistently applied across the business.

  Without a common security infrastructure, Web content and application security policies tend to be applied differently by various parts of the business. This results in a hodge-podge of differing security mechanisms that enforce policy in different ways, often to the point where one cannot easily understand what the organization's overall security policies are.

- The costs of Web security management must be predictable.

  Security requirements evolve with the business. Ultimately, the costs of a commonly leveraged solution that is reliable and scalable to the needs of the business will be far more predictable than other approaches.

- Threats of inadvertent security compromises or hacker attacks represent significant risks to business operations and company goodwill.

  The direct costs of investigation and recovery after a security incident may be significant, but the indirect costs may be even greater. Especially when doing business on the Web, a perception that security is inconsistent and may be compromised can cause substantial revenue loss.

- Competitors are leveraging security solutions to explicitly generate user trust.

  Even if threats are minimal, it still may be essential to maximize the trust that users have in the business's ability to protect itself from compromise. Competitors who can successfully present a solid, secure image may have an advantage over a business that does not.

6.2.2 Typical design objectives (technical requirements)

In conjunction with the business requirements that drive the need for a Web security solution, the following design objectives (technical requirements) are often encountered:

- There is a need to apply security policy independent of application logic.
- A common security control point for Web infrastructure is needed.
- Security policy management must be operating system platform independent.
- Single sign-on (SSO) for access to Web content and applications is needed.

- Authorization policy management and enforcement mechanisms must be consistent across applications.
- Exposure of Web content and applications to potential attack must be minimized.
- There must be a common audit trail of accesses to all Web applications.

These are only examples of some of the possible design objectives that might drive Web security solutions, such as those utilizing WebSEAL. Applying common security architecture subsystems and network models 2.1, “Common security architecture subsystems” on page 20 to individual scenarios will generate fine-grained design objectives that can be applied within the solution.

6.3 Web security architectural principles

The most common Access Manager scenarios involve management of access to Web content using WebSEAL and the Web Plug-in. Our approach to these architectures is based on three principles, consistently applied.

6.3.1 Principle 1

Web security must begin at the front gate.

This means that, first, there should be a logical Web “front gate” to your content and applications. Side and back gates create vulnerabilities. Second, you must control access at this point, because after someone gets inside, there are many more available channels through which vulnerabilities may be exploited. Your Web front gate is also the initial “choke point” for auditing access attempts.

WebSEAL is the Access Manager component that provides this logical Web front gate. Its authentication capabilities and integration with the Access Manager authorization services enable us to know who a user is and make appropriate access decisions before exposing any additional Web infrastructure.

6.3.2 Principle 2

Minimize the number of direct paths to each component.

Ideally, we should have only one HTTP/HTTPS path to our Web servers from a browser. To enforce this, we can utilize the stateful packet filtering capabilities of firewalls to allow or prevent certain traffic.

This can protect us from certain types of attack, unless the firewall itself is compromised. The attacker then may be able to launch a multitude of direct

attacks on the Web server in an attempt to gain direct access to sensitive content and control of applications. By interposing a reverse proxy such as WebSEAL, the range of possible attack scenarios in the event of a firewall compromise is lessened.

6.3.3 Principle 3

Keep critical content and application functions away from hosts that directly interface to Web clients (that is, browsers).

The further away components are from a potential attacker, the easier it is to minimize the number of available direct paths to exploit them.

6.4 Access Manager for e-business components

In addition to the core components of Access Manager described in Chapter 5, "Access Manager core components" on page 163, Access Manager for e-business has several resource managers that build upon the core infrastructure to provide access control to Web-based applications. These resource managers are described in this section.

6.4.1 WebSEAL

WebSEAL is a high-performance, multi-threaded reverse proxy, that sits in front of back-end Web services. It applies a security policy to a protected object space (which is defined in the authorization database, described in 5.3.3, "Authorization database" on page 177). WebSEAL can provide SSO solutions and incorporate back-end Web application server resources into its security policy. Being implemented on an HTTP server foundation, it listens to the typical HTTP and HTTPS ports.

More details about positioning Access Manager components, especially WebSEAL, within an Internet-centric environment can be found in 6.5, "Basic WebSEAL component interactions" on page 215.

Junctions

The back-end services to which WebSEAL can proxy are defined via *junctions*, which define a set of one or more back-end Web servers that are associated with a particular URL.

Traditional junctions

Traditional WebSEAL junctions are created by defining a new point in the URI space that indicates to WebSEAL which server to direct the request to.

For example, suppose a junction on the WebSEAL host *www.abc.com* is defined such that a request for any URL specifying the path */content/xyz* (relative to the Web space root, of course) is to be proxied to the back-end Web server *def.internal.abc.com*. /content/xyz is the *junction point*, which can be thought of in a loose sense as being similar in concept to a file system mount point.

A user at a browser then makes a request for *http://www.abc.com/content/xyz/myhtmlfiles/test.html*. WebSEAL examines the URL and determines whether the request falls within the Web space for the /content/xyz junction point. It then proxies the request to def.internal.abc.com and forwards the resulting response back to the browser.

From the perspective of the browser, the request is processed by www.abc.com. The fact that it is actually handled by the target server def.internal.abc.com is not known to the user. To support this, WebSEAL performs various transformations on the response sent to the browser to assure that the back-end server names are not exposed. This exemplifies one of the powerful capabilities provided by WebSEAL junctions (that is, the “virtualization of the Web space”). Junctions may be defined to the individual Web spaces on various back-end servers, yet from the browser’s point of view, there is only one single Web space.

Figure 6-2 illustrates traditional WebSEAL junctions.

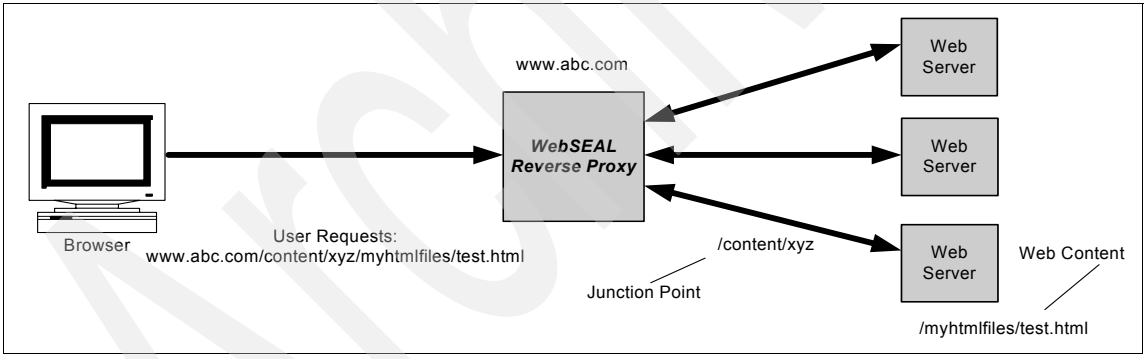


Figure 6-2 WebSEAL traditional junctions

With the above scenario, WebSEAL is surveying all Web traffic to ensure that the junction name is included in the request, constantly filtering all HTTP responses to ensure that all links are properly constructed. The junction name must be added to every Web address to ensure proper routing of the requests. In simple Web environments, this is not an issue. However, in existing and complex environments, alterations become much more complex. Thus the need for a non-invasive method of supplying connections to Web servers becomes apparent.

Virtual host junctions

WebSEAL provides two methods for accomplishing this through the use of Virtual Host junctions and transparent path junctions. Virtual Host junctions preserve the traditional Web addresses that may already exist within a corporation. For example, a company may have www.myhr.com for their HR system and www.mypayroll.com for their payroll system. Since these applications already exist and their Web addresses are known throughout the user community, the application of the traditional WebSEAL junction method would not benefit the corporation. Instead, resolving www.myhr.com and www.mypayroll.com to WebSEAL's IP address and allowing it to decipher which server to direct traffic to would be the most beneficial.

Figure 6-3 illustrates Virtual Host junctions.

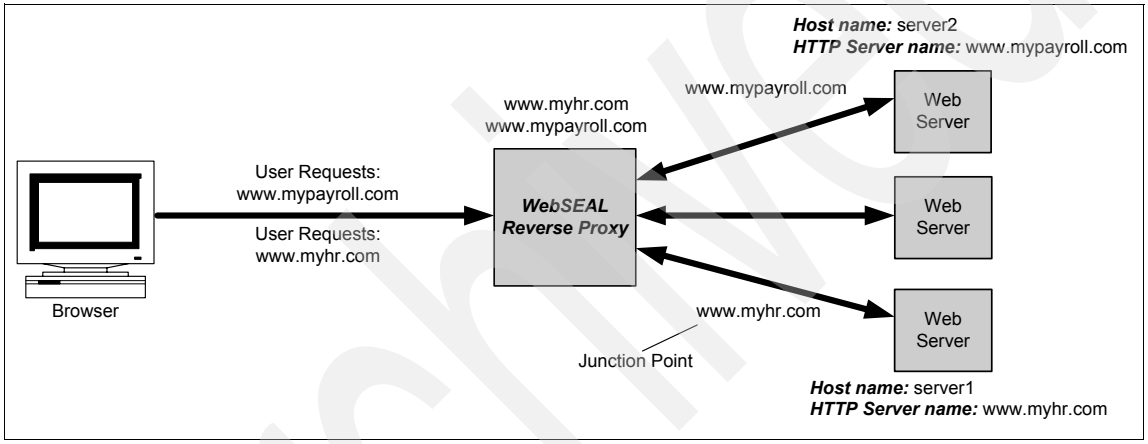


Figure 6-3 WebSEAL Virtual Host junctions

In the above scenario, no filtering of content transmitted from the backend server is performed or required.

Note: Virtual Host junctions may only be used if the same protocol is used throughout the entire transaction. The connection from the browser to WebSEAL must be over the same protocol and port as the connection from WebSEAL to the backend server.

Also, Virtual Host junctions introduce unique challenges for performing SSO and session management which are discussed later.

Transparent path junctions

In order to combine the benefits of both a single URL space for session management and SSO without the problems of path filtering, Access Manager for e-business is using the concept of *transparent path junctions*. Transparent path junctions remove the need for the junction name such as /content/xyz to be included in the Web address. Instead, transparent path junctions are part of the existing URI space located on the backend server. In the example of www.abc.com, the transparent junction would simply be /myhtmlfiles. There is no need to add an extra junction name.

Figure 6-4 illustrates transparent path junctions.

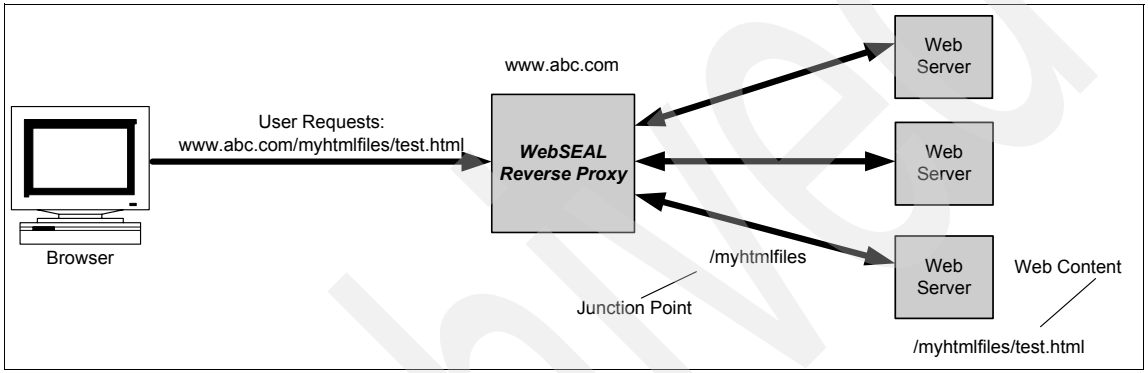


Figure 6-4 WebSEAL transparent path junctions

WebSEAL does not need to filter server relative links in HTTP responses like it does with traditional junctions. WebSEAL simply matches part of the URI after the WebSEAL address. Filtering is still required on all absolute URLs to ensure they point to WebSEAL and not the backend server.

Note: Transparent path junctions require that the backend URI exists on only one server. For example, server 1 has a backend URI space of /path1/xxx and server 2 has a backend URI space of /path1/yyy. This will cause a problem in terms of transparent path junctions if the junction was equal to /path1. To resolve this, it is necessary to further define the junction down to a level in the URI space that is unique. For our example junctions of /path1/xxx and /path1/yyy would be sufficient. However, if both servers had identical URI spaces it would only be possible to use transparent path junctions if two distinct WebSEALs were used in order to avoid conflicts. It is best in this type of situation to consider using virtual host junctions.

Common junction features

More than one target server may be defined for a given junction point. For example, the server ghi.internal.abc.com could be defined as an additional target for the /content/xyz junction point. In this case, WebSEAL can load-balance among the servers, and if a back-end server is unavailable, WebSEAL can continue forwarding requests to the remaining servers for the junction. Load-balancing is available for all WebSEAL junctions including traditional, virtual host, and transparent.

WebSEAL can also throttle requests and turn off requests to all servers on a particular junction or an individual server. This enables a server to be taken out of the Web space for maintenance without affecting end users.

For situations in which it is important that subsequent requests for a particular user continue going to the same back-end server, WebSEAL is capable of supporting this via what are called *stateful junctions*. By default, WebSEAL will always route requests to the same stateful junction server, even if that system fails, but it is possible to configure WebSEAL to route requests to a new stateful junction server if the original fails. See Chapter 8, “Increasing availability and scalability” on page 259, for more information about high-availability with junctions.

WebSEAL security functions

One of WebSEAL’s key functions is to protect access to Web content and applications. To do this, it uses Access Manager’s Authorization Services. The Authorization Service must know which Web objects (that is, URLs) require protection, and what level(s) of access to these objects is permitted for the Access Manager users and groups defined in the user registry.

The protected object space is defined in the Access Manager authorization database. It can be populated using a special CGI program that runs on each back-end junctioned Web server. This program, named query_contents, is run by the Web Portal Manager and scans the Web directory hierarchy on the server. It populates the authorization database with representations of the various objects it finds. ACLs, POPs, and authorization rules can be “attached” to these objects, and WebSEAL can then use Access Manager’s authorization engine to make access decisions about requests for various URLs.

Note: Query_contents can be customized to deal with different application types on different operating platforms, and it is not necessary for the object space to be populated in order to attach policy to objects. The population is for presentation convenience, it is not a prerequisite for being able to apply policy.

Of course, the authorization engine cannot make access decisions without being told something about the identity of the user. WebSEAL supports the ability to authenticate a user and assign an Access Manager identity for use when making authorization decisions. Whenever a URL is requested that is not accessible by an unauthenticated user, WebSEAL attempts to authenticate the user by issuing an authentication challenge to the browser (it supports multiple authentication mechanisms, which are discussed in section 9.4, "Web security server authentication mechanisms" on page 291). Upon establishment of an authenticated "session," the authorization engine is then consulted to determine whether the user may access the content specified by the requested URL. This WebSEAL session is maintained until the user exits the browser or explicitly logs off, or until the session times out or is terminated by an administrator. Subsequent URL requests for this session continue to be checked to determine whether access is permitted.

The access control granularity that is provided can range from a coarse-grained protection of particular directories (containers) in the Web space to specific, fine-grained protection of individual Web objects (for example, an individual HTML file). Additionally, URL "patterns" may be defined that represent dynamic URLs. For example, application parameters are often defined in URLs and may differ across invocations. By defining a pattern to Access Manager's Web object space that matches such a URL, it is possible to accommodate these situations.

Administrative support

As an added functionality, WebSEAL supports a *switch user* function. It enables administrators to log on to Access Manager as a user without having to supply a password. This aids help desk administrators with customer support issues. It can also be used by administrators to easily troubleshoot and verify the correct functionality of access control lists without the need to create test users.

Authentication to back-end servers

Often it is necessary to provide special authentication information to junctioned Web servers to verify the identity of the WebSEAL server, provide the identity of the logged-in user, or both. WebSEAL provides a number of mechanisms to support such authentication requirements. This is the typical representation for SSO, and more information can be found in section 9.3.1, "Authentication and single sign-on mechanisms" on page 284.

WebSEAL authentication

If necessary, WebSEAL can authenticate itself to a junctioned server using either server certificates, forms-based authentication, HTTP basic authentication, or by sending its server name in configurable HTTP headers. When using a Secure Sockets Layer (SSL) communication channel for this junction, WebSEAL and the junctioned server can also mutually authenticate each other. This is very

important in order to establish the trust relationships between WebSEAL and back-end Web application servers.

Single sign-on

WebSEAL supports several mechanisms for supplying a junctioned server with the identity of the logged-in user, including:

- Providing the user's identity via HTTP header values, which can be read and interpreted by the junctioned server.
- Insertion of an HTTP basic authentication header to provide the junctioned server with login information for the user, including a password. Optionally, this basic authentication header can permit login to the junctioned server with a different identity from the one for the user who is logged in to WebSEAL.
- For junctions that support it (for example, WebSphere Application Server and Domino), insert a *Lightweight Third-Party Authentication* (LTPA) cookie identifying the user into the HTTP stream that is passed to the junctioned server.
- For junctions that support it, (WebSphere Application Server), the use of a *Trust Association Interceptor* Plus (TAI++) to forward Tivoli Access Manager credential information and establish trust between WebSEAL and backend application server.

WebSEAL-delegated authentication capabilities are discussed more in 9.5, "Web security server single sign-on mechanisms" on page 306.

Replicated WebSEALs

It is possible to replicate WebSEAL servers for availability and scalability purposes. There are specific configuration requirements for creating WebSEAL replicas, and a front-end load balancing capability must be used to distribute incoming requests among the replicas. Also, since each WebSEAL replica, by default, maintains active session states for its own authenticated users, it is recommended that the Session Management Server (SMS) be used to maintain state and avoid limitations for policy enforcement, management, security, and the user experience. The Session Management Server is described in more detail in 6.4.6, "Access Manager Session Management Server" on page 210. The use of WebSEAL replicas is discussed and illustrated in Chapter 8, "Increasing availability and scalability" on page 259.

Virtual hosting

Multiple instances of WebSEAL can be created on a single machine using the WebSEAL configuration/unconfiguration utility. Also, a single WebSEAL instance can listen to multiple interfaces and multiple ports. Different IP and SSL

configuration information can be associated with each interface. This is necessary to support Virtual Host Junctions.

Note: Even though it is possible for WebSEAL to support multiple DNS names, the functionality is only intended for use in conjunction with virtual host junctions. See “Virtual host junctions” on page 198 for more information.

Communication protocols

WebSEAL can communicate with the clients and back-end servers with both encrypted (HTTPS) and unencrypted (HTTP) protocols. The supported encryption types are SSLv1, SSLv2, SSLv3, and TLSv1.

Secure Sockets Layer hardware acceleration support

For performance improvement, WebSEAL supports SSL hardware acceleration. Utilizing the functionality of GSKit7, hardware acceleration can minimize the CPU impact of SSL communications, improving the overall performance of the system.

This support applies to any SSL session that WebSEAL is involved in, but the performance impact visible to users is exclusive to the browser-WebSEAL session. The performance advantage provided by the SSL hardware acceleration card is the initial SSL handshaking between two communicating parties. When an SSL tunnel is set up, the card does not help any more. In other words, the card provides benefits only for the RSA or PKCS11 public key authentication part (happening in the initial SSL handshaking), but not for the DES encryption part used in normal data transmission afterwards. Some SSL sessions are built during the configuration time or the junction setup time and will be reused, so we will not see performance improvement from SSL hardware acceleration for these sessions. The browser-to-WebSEAL SSL session is built whenever a browser first connects to WebSEAL. The customer value is the improved performance in browser-WebSEAL SSL session setups and the higher numbers of users who can be supported due to the off-loading of work from the WebSEAL host’s processor to the card. For more information about SSL hardware acceleration support, look “Cryptographic hardware for encryption and key storage” in the *IBM Tivoli Access Manager for e-business Version 6.0 WebSEAL Administration Guide,* SC32-1687.

Other WebSEAL functionality

WebSEAL supports an e-community SSO functionality that enables Web users to perform an SSO and move seamlessly between WebSEAL servers in two separate secure domains.

WebSEAL also supports a capability that permits failover of logged-on users to another replica in the same domain in the event that their assigned WebSEAL

server becomes unavailable. This failover cookie feature is also supported by the Plug-In for Edge Server, which is discussed in 6.4.5, “Plug-in for Edge Server” on page 209.

Architecture

The WebSEAL architecture is summarized in Figure 6-5. The WebSEAL server directly interacts with the browser and proxies requests to junctioned Web servers, determining which junction to pass the request to by examining various components of the HTTP request.

Before passing the request, WebSEAL also uses the authorization engine to check the URL against the Web objects. If the URL is not protected, the request is simply proxied to the appropriate junction. If the URL is protected, an access control check must first be made. If the user is not yet authenticated, an authentication challenge is sent to the browser, and WebSEAL uses its authentication services to validate the user’s claimed identity and map it to an appropriate Access Manager identity in the user registry. Access to the object is then checked against this identity and, if allowed, the request is proxied.

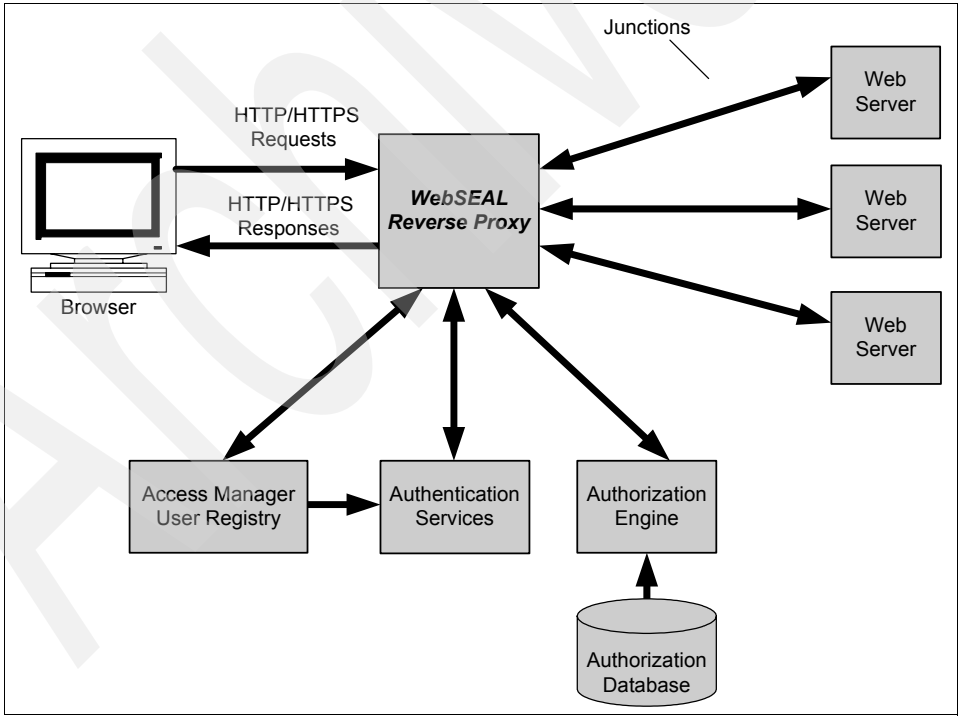


Figure 6-5 WebSEAL architecture

6.4.2 Plug-in for Web servers

The Web server plug-in architecture provides a solution where the customer has decided to deploy a Web plug-in architecture for his solution architecture rather than a reverse proxy approach.

Table 6-1 summarizes the capabilities that are provided by this implementation based on a WebSEAL comparison.

Table 6-1 Plug-In for Web server functionalities

Authentication support	
Authentication based on client IP address	Supported
User name/password (basic authentication and forms-based), certificate, and SecureID authentication	Supported
Step-up authentication	Supported
External authentication C API	Supported
Interoperability with WebSEAL failover cookies	Supported
Web SSO (basic authentication and forms-based authentication)	Supported
SSO from plug-in Web server to back-end BEA WebLogic Server (BEA WLS) or WebSphere Application Server	Supported
e-community SSO (requires a WebSEAL Master Authentication Server (MAS)	Supported
SSO from WebSEAL to plug-in	Supported
Forms-based SSO	Supported
Password policy support, including password strength, password expiration, extensible password policy native implementation of “N strikes out password policy”	Supported
Junction from WebSEAL to plug-in	Supported; will accept WebSEAL-to-WebSEAL junctions
Authorization support	
ACL and POP policies	Supported
Tag/value support	Supported

Authentication support	
PD_PORTAL support	Supported
Pass user/groups/creds in HTTP header	Supported
Failover (same as authentication failover)	Supported
Platform support	
Note: One Web server per host is assumed.	
IIS 6.0	► IIS 6.0 ► Windows 2003 Server (Not supported on Windows 2003 Datacenter)
Sun ONE Web server 6.0	► Sun ONE Web server 6.0 SP7 ► Solaris 8 and 9 (SPARC) ► AIX 5.1 and 5.2
Sun Java System Web server	► Sun Java System Web server 6 ► Solaris 8 and 9 (SPARC) ► AIX 5.1 and 5.2
Apache 1.3.27	► Apache 1.3.27 ► Solaris 8 and 9 (SPARC) ► Linux on zSeries (Red Hat 3 and 4 or SLES 8 and 9)
Apache 2.0.48	► Apache 2.0.48 ► AIX 5.2 ► Solaris 10 ► Linux on zSeries (Red Hat 3 and 4 or SLES 9)
IBM HTTP Server 1.3.26	► IBM HTTP Server 1.3.26 ► AIX 5.1 and AIX 5.2 ► Solaris 8 and 9 (SPARC) ► Linux Intel (Red Hat 3 and 4 or SLES 8) ► Linux on zSeries (Red Hat 3 and 4 or SLES 8)

Authentication support	
IBM HTTP Server 2.0.47	► AIX 5.2 and 5.3 ► Solaris 10 (SPARC) ► Linux Intel (Red Hat 3 and 4 or SLES 9) ► Linux on zSeries (Red Hat 3 and 4 or SLES 9)
Other	
Directory support: IBM Tivoli Directory Server, Sun Java System Directory, Microsoft Active Directory, and Lotus Domino	Supported
Virtual hosting	Provided by the host Web server
Install/configure/uninstall	Simple and intuitive
URL and HTTP protocol transparency	Supported
Globalization-languages supported	Same as Access Manager Base, except bi-directional languages will not be supported

Figure 6-6 shows an architectural overview of the Web server plug-in implementation.

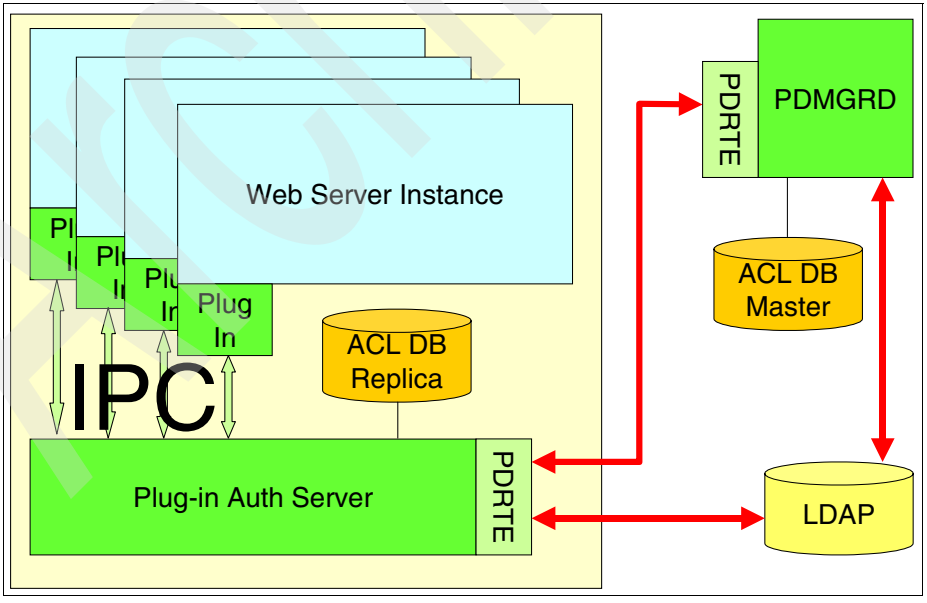


Figure 6-6 Access Manager Web server plug-in architecture

In most Web server environments, there are multiple server threads in operation on the machine. These might be different threads of the same Web server instance or threads of different Web server instances. Having a distinct authorization engine for each thread would be inefficient, but would also mean that session information would have to be shared between them somehow.

The architecture used contains two parts:

- Interceptor

  This is the real *plug-in* part of the solution. Each Web server thread has a plug-in running in it that gets to see and handle each request/response that the thread deals with. The interceptor does not authorize the decisions itself; it sends details of each request (via an interprocess communication interface) to the Plug-In Authorization Server.

- Plug-In Authorization Server

  This is where authorization decisions are made and the action to be taken is decided. There is a single Plug-In Authorization Server on each machine and it can handle requests from all plug-in types. The Plug-In Authorization Server is a local aznAPI application that handles authentication and authorization for the plug-ins. The Authorization Server receives intercepted requests from the plug-ins and responds with a set of commands that tell the plug-in how to handle the request.

6.4.3 Access Manager Attribute Retrieval Service

The Access Manager Attribute Retrieval Service allows for Authorization Rules (Rules) to be written that require Authorization Decision Information (ADI) that is not available in any information that the Tivoli Access Manager authorization service has access to. This retrieval can be performed real-time by a dynamic ADI entitlement retrieval service. The attribute retrieval service currently provided with WebSEAL is one type of entitlement retrieval service.

6.4.4 Common Auditing and Reporting Service

IBM Tivoli Access Manager for e-business 6.0 has the ability to utilize the IBM Tivoli Common Auditing and Reporting Service to collect audit data in a central location, run reports against the audit data, and archive audit data. More information about the Common Auditing and Reporting Service is in Chapter 27, “Introducing IBM Tivoli Common Auditing and Reporting Service” on page 845.

6.4.5 Plug-in for Edge Server

The Access Manager Plug-In for Edge Server is a plug-in for the Edge Server Caching Proxy component of the IBM WebSphere Edge Server. It adds Access Manager authentication and authorization capabilities to the proxy, and in certain scenarios it provides an alternative to WebSEAL for managing access to Web content and applications.

While the Plug-In for Edge Server shares many of the same capabilities as WebSEAL, its configuration is different. However, architecturally, it fits into most Access Manager scenarios in the same manner as WebSEAL.

Among other differences are two key differentiators between the plug-in and WebSEAL:

- Use of the plug-in with the Edge Server Caching Proxy provides direct support for virtual hosting.
- The plug-in can be used in both forward and reverse proxy configurations (WebSEAL only supports a reverse proxy).

The plug-in also integrates with the WebSphere Everyplace® Suite and supports forms-based login and Access Manager WebSEAL failover cookies.

Architecture

Figure 6-7 on page 210 provides a simplified view of the Plug-In for Edge Server architecture used as a reverse proxy (a forward proxy scenario is virtually identical, except that the proxy operations are to the outside rather than back-end servers). It should be noted that while this architecture is similar to that for WebSEAL (Figure 6-5 on page 204), the specific functionality and configuration of various components does differ.

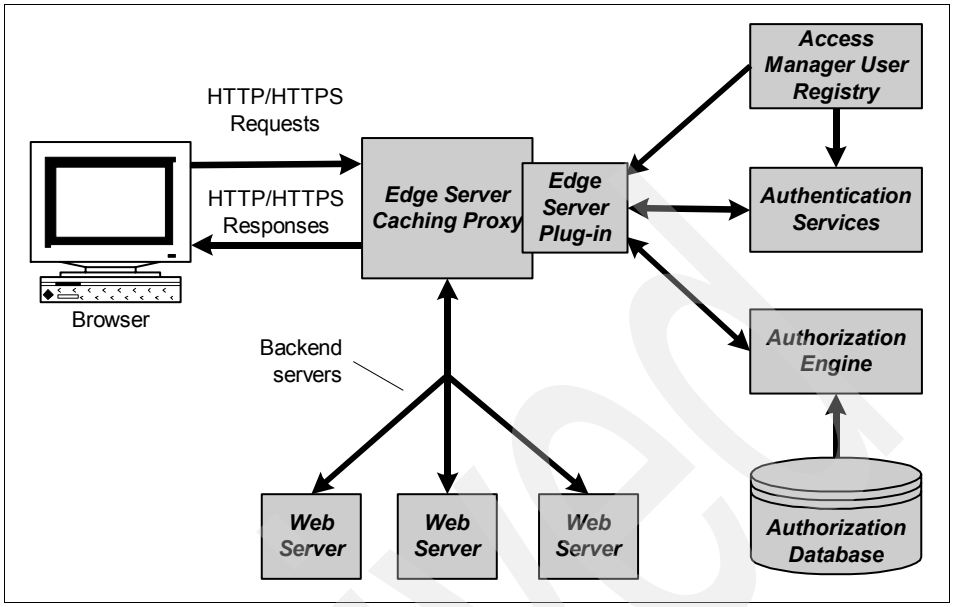


Figure 6-7 Plug-in for Edge Server architecture

6.4.6 Access Manager Session Management Server

Access Manager Session Management Server (SMS) is an optional Tivoli Access Manager component that runs as an IBM WebSphere Application Server service. It manages user sessions across complex clusters of Tivoli Access Manager security servers, ensuring that session policy remains consistent across the participating servers. Using the Session Management Server allows Access Manager WebSEAL and Access Manager Plug-in for Web servers to share a unified view of all current sessions and permits an authorized user to monitor and administer user sessions. The Session Management Server permits the sharing of session information and also makes available session statistics and provides secure and high-performance failover and SSO capabilities for clustered environments.

The Session Management Server provides a user interface from which authorized persons can administer and monitor user sessions. Administration of the Session Management Server is performed using either the pdadmin command line utility or using the Session Management Server Web-based graphical user interface that is run from within the Access Manager Web Portal Manager.

Figure 6-8 on page 211 shows how multiple security servers can achieve a single session by using a common Session Management Server that provides a unified

backing store for session data. Each Web security server maintains a local copy of the session data in its own session cache for performance reasons. A backup or master copy is also maintained on the Session Management Server and this data can be accessed by other Web security servers when necessary. The Web security servers work with the Session Management Server to create, retrieve, and update the shared session data. The Session Management Server provides updates to Web servers that are participating in a given user session - alerting them to urgent changes in the session data such as a user logging out.

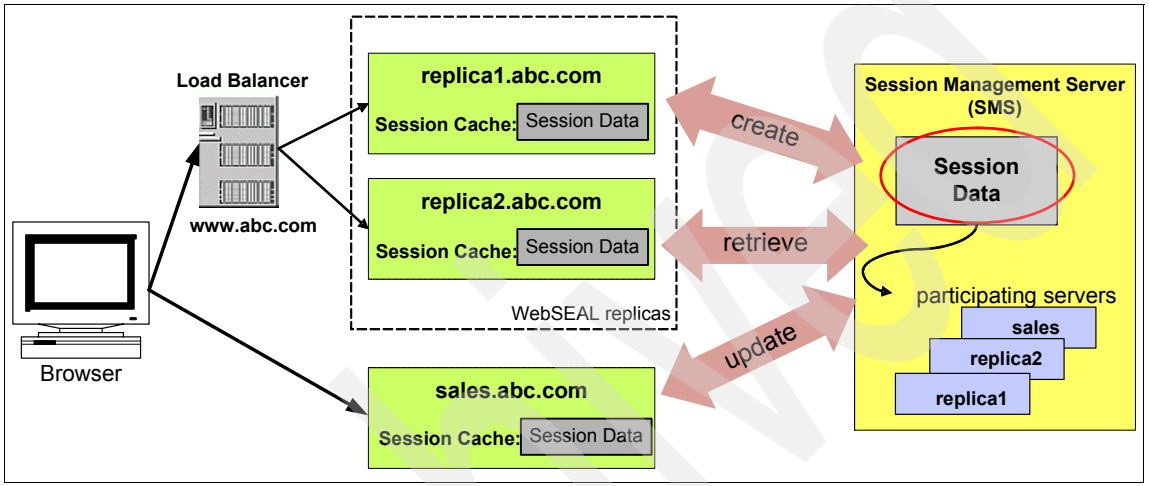


Figure 6-8 Access Manager Session Management Server

Through the use of the Session Management Server, it is now possible to present a consistent user experience across all Web security servers as well as providing the ability to strictly enforce security policy such as maximum number of sessions.

6.4.7 Access Manager for Microsoft .NET applications

Tivoli Access Manager exposes the APIs at the .NET Common Language Runtime (CLR) level. This allows Access Manager functionality to be available to all .NET languages such as Managed C++, C#, and Visual Basic .NET.

Access Manager for Microsoft .NET provides SSO from Tivoli Access Manager Web security servers (WebSEAL and plug-in for Web servers) to ASP .NET applications. Put simply, the .NET application can accept an Access Manager user ID or credential and authenticate traffic origin.

Figure 6-9 illustrates how Access Manager provides SSO in a Microsoft .NET environment.

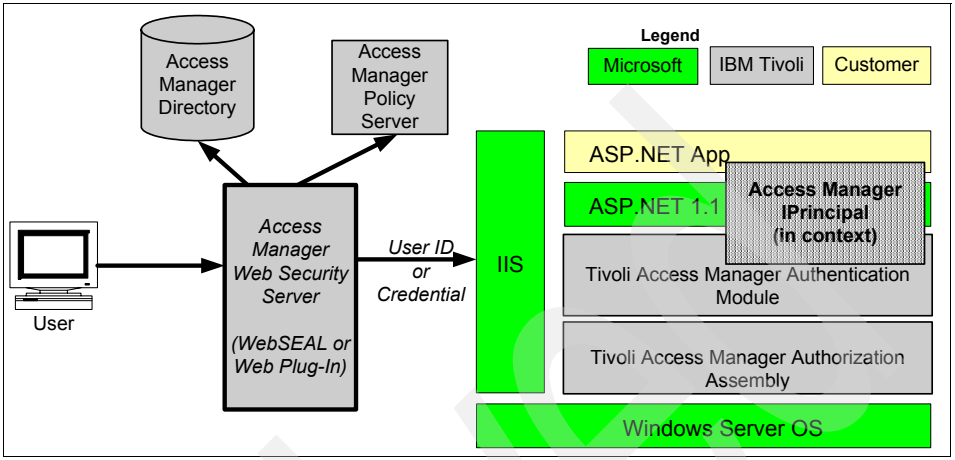


Figure 6-9 Access Manager for .NET SSO

In addition, role membership is evaluated using Tivoli Access Manager policy in one of two ways:

- Declarative role security where the ASP .NET container enforces roles declared by the application
- Programmatic role security where the application makes an API call to determine whether a user possesses a particular role.

No changes are required to any code in order to use Access Manager authorization provided that the application is using either the declarative security model or the programmatic security model. Access Manager will use one of two approaches to determine if the user possesses a given role:

- User-to-role mapping via the user's group membership
- User-to-role mapping via an Access Manager authorization check of an object in the Access Manager protected objectspace that represents the role

Note: While the user-to-role mapping via group membership is the simpler of the two models, it does have some limitations. Advanced authorization policy such as Protected Object Policies (POPs) and Authorization Rules (Rules) cannot be used. Also, any change to a policy will not be effective until the next time the user logs in. If a more advanced and dynamic security policy is required, the user-to-role mapping via an Access Manager authorization check should be used.

Access Manager for Microsoft .NET also provides for Web services security in one of two ways:

- Client-side authorization and identity propagation via HTTP headers
- Server-side authentication and authorization via HTTP header or SOAP WS-Security header (Username Token)

There are two APIs that are exposed to .NET applications:

- .NET Assembly for Tivoli Access Manager Administration Services
- .NET Assembly for Tivoli Access Manager Authorization Services

Access Manager for Microsoft .NET allows for a user to change their role dynamically without restarting the user's session or the application. In addition, Access Manager can use any directory for the security information that is supported by the core components. More details can be obtained in 11.5, "Access Manager and .NET Integration" on page 367.

6.4.8 WebSphere Application Server integration

Starting with WebSphere Application Server 5.1.1 and above, WebSphere Application Server ships with all the Access Manager Java Runtime Environment and .jar files required for integration into an Access Manager secure domain. This is not a separate product, but an integration point between Access Manager and WebSphere that can be used to centralize security for J2EE applications in one location using Access Manager. In addition, a J2EE-to-Access Manager user/role migration utility is provided to assist customers in populating the Access Manager policy database with users and roles.

This enables enterprises to leverage a common security model across WebSphere and non-WebSphere resources leveraging common user identity and profiles, Access Manager-based authorization, and using Access Manager's Web Portal Manager to leverage a single point of security management across J2EE and non-J2EE resources.

The integration is transparent to the J2EE applications because no coding or deployment changes are needed at the application level. More details can be obtained in 11.3, "WebSphere Application Server security" on page 352 and 11.4, "Access Manager and WebSphere integration" on page 357.

6.4.9 Access Manager for BEA WebLogic Server

Tivoli Access Manager for WebLogic Version provides a full security framework for BEA WebLogic Server using the Security Service Provider Interface (SSPI).

BEA WebLogic Server provides SSPI for third-party security providers, such as Tivoli Access Manager for WebLogic, to seamlessly integrate their security functions into the BEA WebLogic Server architecture.

Note: Check the *IBM Tivoli Access Manager for e-business Version 6.0 Release Notes,* SC32-1702 for the supported version(s) of the BEA WebLogic Server.

Access Manager Security Service Provider Interface components

Tivoli Access Manager for WebLogic replaces the default security realm created with each BEA WebLogic Server secure domain and provides the following BEA WebLogic Server Security Providers:

- Authentication Provider
- Authorization Provider
- Role Mapping Provider

Tivoli Access Manager for WebLogic uses the default BEA WebLogic Server Credential Mapping security provider and the default keystore.

Each of the providers listed above also contains a Management Bean (MBean) that enables configuration editing through the WebLogic console. The sections below detail the functionality supplied by each of these providers and MBeans.

Tivoli Access Manager provides the following integration points with BEA WebLogic Server:

- Authentication Provider

  The Tivoli Access Manager for WebLogic Authentication Provider implements BEA WebLogic Server simple authentication. In simple authentication, a user attempts to authenticate to a BEA WebLogic Server with a user name and password combination. This user name and password are checked by Tivoli Access Manager using the Tivoli Access Manager Java runtime component.

  Tivoli Access Manager for WebLogic also provides its own Login Module that is used to provide WebSEAL or Tivoli Access Manager Plug-in for Web servers SSO functionality.

- Authorization Provider

  Authorization Providers supply an interface between BEA WebLogic Server and the external authorization service. The Authorization Provider determines

whether access should be granted or denied to BEA WebLogic Server resources. The access decision is made using the PDPermission classes that are distributed with the Tivoli Access Manager Java runtime component.

- Role Mapping Provider

  Role Mapping Providers are used to supply an interface between BEA WebLogic Server and the external authorization service that is being used to manage roles. The Role Mapping Provider focuses on roles rather than on policy, which is the responsibility of the Authorization Provider.

Policy and role deployment

Policy and roles can be defined in deployment descriptors or created through the WebLogic console. Upon deployment of J2EE applications, roles and policy defined within the application deployment descriptors are exported to the Tivoli Access Manager protected object space.

Although possible, it is not expected that policy creation will be performed using the Tivoli Access Manager administrative utility, pdadmin, or the Tivoli Access Manager Web Portal Manager. Before starting a BEA WebLogic Server that is using Tivoli Access Manager for WebLogic, some default policy must be created in Tivoli Access Manager. This is performed during configuration of Tivoli Access Manager for WebLogic.

Resources and roles

BEA WebLogic Server defines a number of different resource types, all of which are supported by Tivoli Access Manager for WebLogic. All resource types are considered the same within Tivoli Access Manager for WebLogic, so new resource types, created for future releases of BEA WebLogic Server, will be supported automatically.

The policies and roles defined for all resource types are stored in the Tivoli Access Manager protected object space in a uniform way.

More information about the BEA WebLogic integration can be found in the *IBM Tivoli Access Manager for e-business BEA WebLogic Server Integration Guide Version 6.0*, SC32-1688.

6.5 Basic WebSEAL component interactions

As discussed in Chapter 5, “Access Manager core components” on page 163, all Access Manager architectures share a common set of base components. Specifically, all Access Manager deployments have a User Registry and a Policy Server. WebSEAL interacts with these components to provide its security functions, as shown in Figure 6-10 on page 216.

The Policy Server, while a part of the overall architecture, is not a constant required component. WebSEAL is designed to function disconnected from the Policy Server, hence the local replica copy of the authorization database. While WebSEAL itself cannot make changes to policy, it can continue to read from its local copy of the authorization database making security decisions until a newly updated copy of the database is received from the Policy Server.

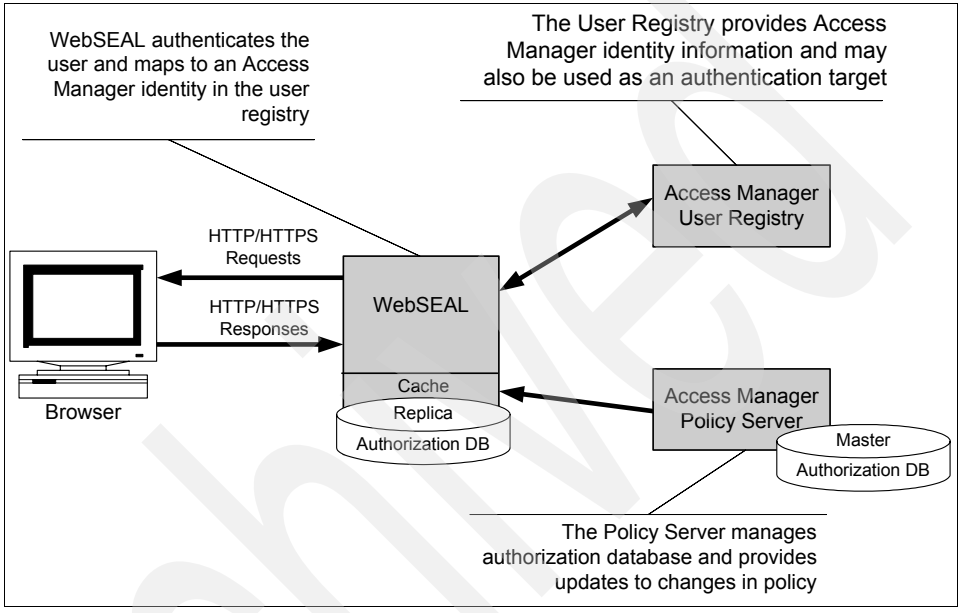


Figure 6-10 WebSEAL interaction with other Access Manager components

In the most basic of WebSEAL architectures, as shown in Figure 6-11, a user at a Web browser contacts WebSEAL with a URL request, and then WebSEAL directly serves the content itself. (Recall that while it functions as a reverse proxy, WebSEAL is also a Web server with the ability to use locally stored content.)

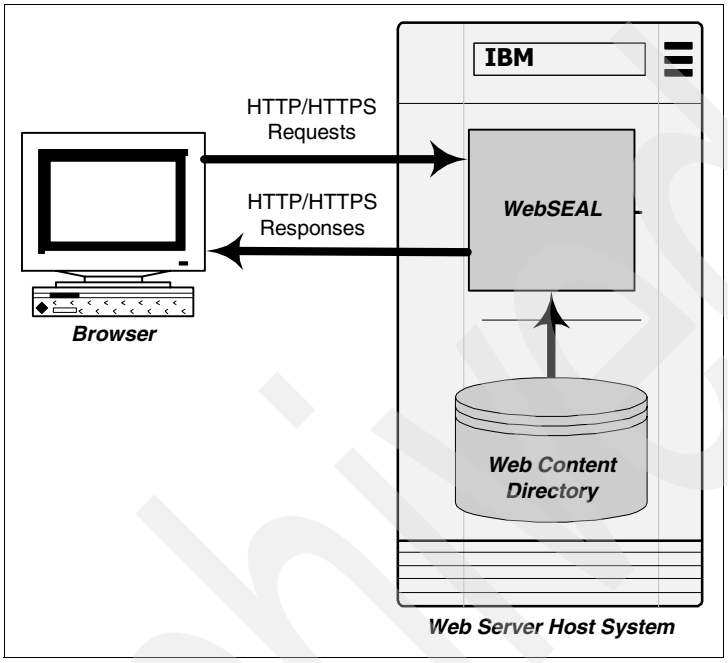


Figure 6-11 Direct serving of Web content from WebSEAL

While illustrative of WebSEAL capabilities, such a scenario may not be terribly interesting, given the evolution of Web-based application architectures that employ significant back-end infrastructure. Also, while directly serving non-sensitive content may be acceptable, when sensitive content is involved, it is generally better to serve it via proxy. Such environments are, in fact, ones where WebSEAL proves to be an ideal solution.

Web applications may involve significant back-end infrastructure, and there are advanced Access Manager scenarios in which direct security integration with such components is important. However, even in complex scenarios, the basic elements of Access Manager architecture still apply.

With WebSEAL junctions, a browser user does not directly interact with the target Web server. Instead, WebSEAL takes care of initial user authentication as required and performs appropriate authorization checks on URL requests. Authorized requests are then proxied via the appropriate junction. Figure 6-12 shows the basic flow involved in processing such a request.

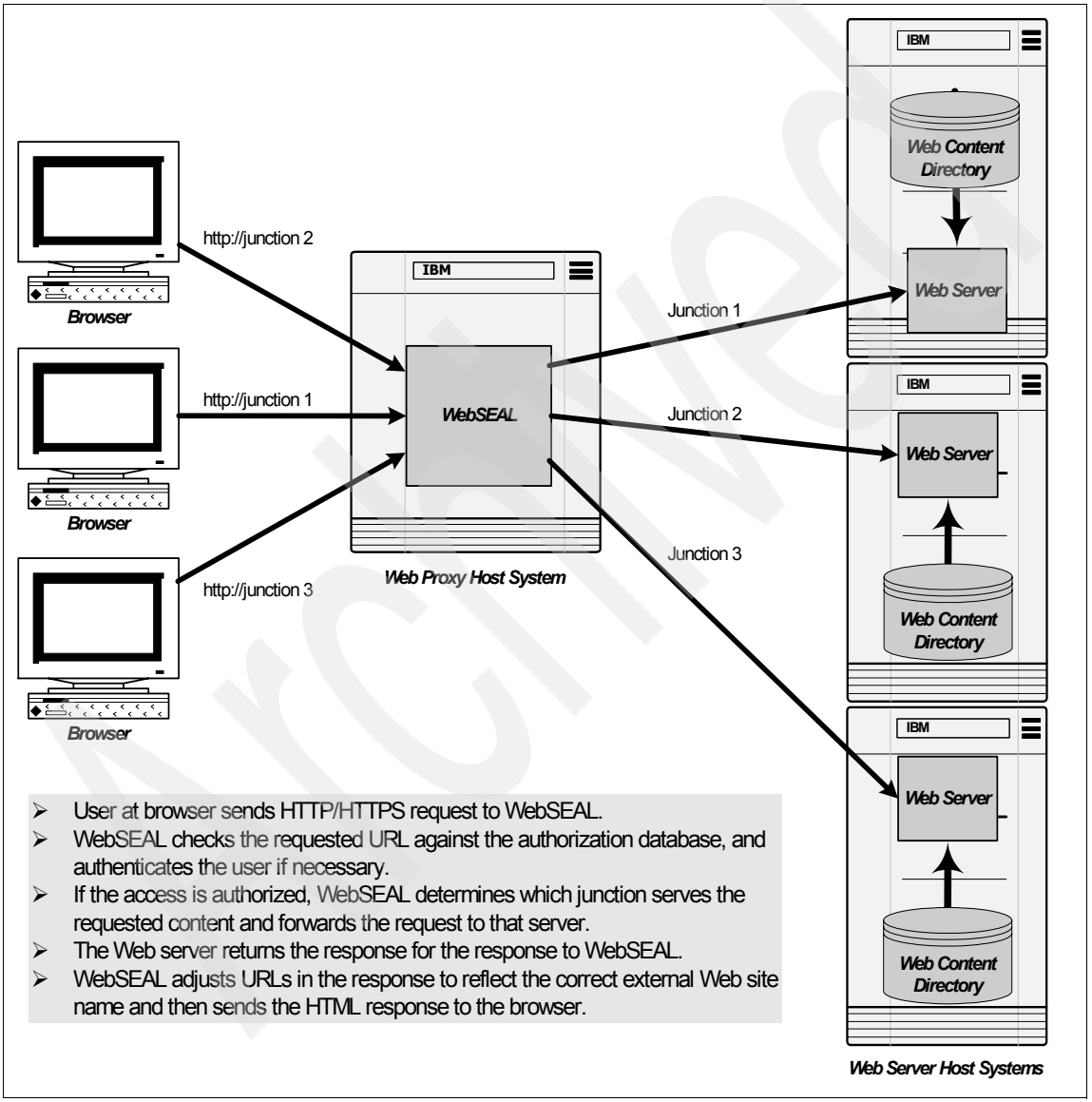


Figure 6-12 Basic WebSEAL proxy functionality

The flow in Figure 6-12 on page 218 represents the common architecture for all WebSEAL deployments. The differences can include the way components are combined or distributed among host systems, junction configurations, and back-end authentication issues. However, WebSEAL deployments are built from the same basic architectural elements.

At this point, we have not yet introduced the role of the network into an Access Manager WebSEAL architecture. Obviously, as we discussed earlier in 6.1, “Typical Internet Web server security characteristics” on page 192, network configuration does play a role, and it is important to understand how WebSEAL and other Access Manager components fit into typical secure network infrastructures.

6.6 Basic Web Plug-in component interaction

Tivoli Access Manager Plug-in for Web servers provides the WebSEAL authentication and authorization capabilities directly to existing Web servers without the need for an additional reverse proxy infrastructure, as is the case with WebSEAL.

The plug-in operates as part of the same process as your Web server, intercepting each request that arrives, determining whether an authorization decision is required, and providing a means for user authentication if necessary. The plug-in can provide SSO solutions and incorporate Web application resources into its security policy.

Two basic architectural components make up Tivoli Access Manager Plug-in for Web servers: the plug-in component and the authorization server. The plug-in component operates with Web server threads sending details of each request, via an Inter-Process Communication (IPC) interface, to the authorization server. The authorization server performs the authentication and authorization of incoming requests. The authorization server is a local mode aznAPI application that accepts and processes requests from the plug-in and responds, telling the plug-in how to handle each request. The component configuration is depicted in Figure 6-13 on page 220.

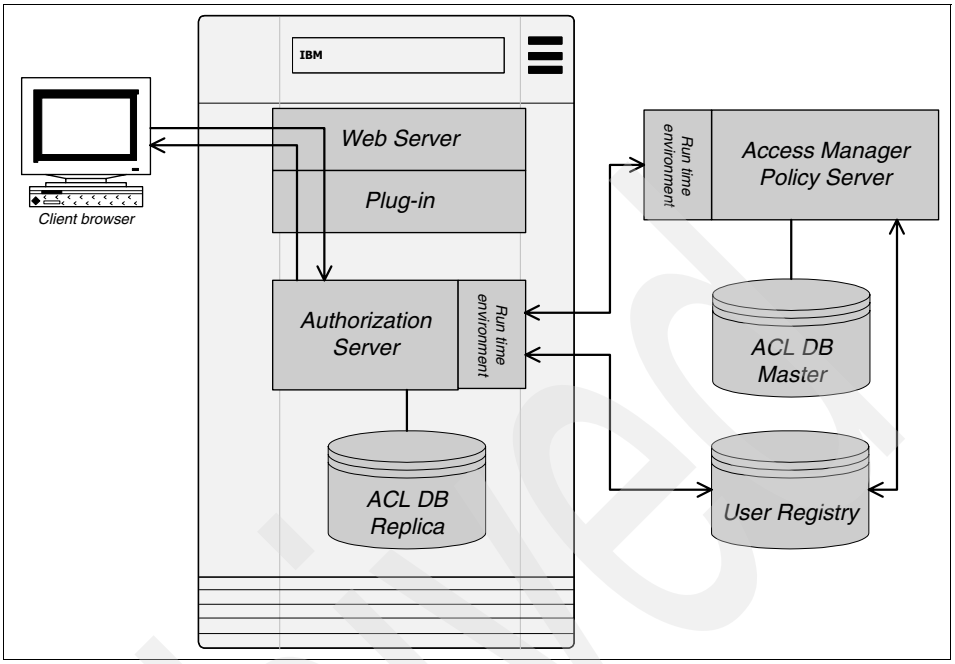


Figure 6-13 Basic Web server plug-in components

The authorization server determines which virtual host the request is addressed to (if virtual hosts are present on the Web server) and whether the request requires authorization. Requests that do not require authorization are passed directly to the Web server for processing. Requests that require authorization are processed by the authorization server in the following way:

1. Session and authentication information is extracted from requests that have been authenticated previously.
2. If required, an authentication interaction is initiated with the user.
3. A Tivoli Access Manager credential is created.
4. The resources the user can access are identified and these resources mapped to the corresponding Tivoli Access Manager protected object name. A protected object name represents an electronic entity such as a secure part of a Web site or an application that only certain users are permitted to access.
5. Whether the request or response requires modification is determined.
6. Responses required by the plug-in or the host Web server are generated by adding cookies or headers to the request or response or by generating a response (for example, an authenticated response or an unauthorized response).

Architecturally, the main difference between the Tivoli Access Manager Web server plug-in and WebSEAL is the lack of reverse proxy capabilities, as shown in Figure 6-14. Tivoli Access Manager for e-business 5.1 provides virtually all of the same authentication and authorization functionality with the plug-in as with WebSEAL.

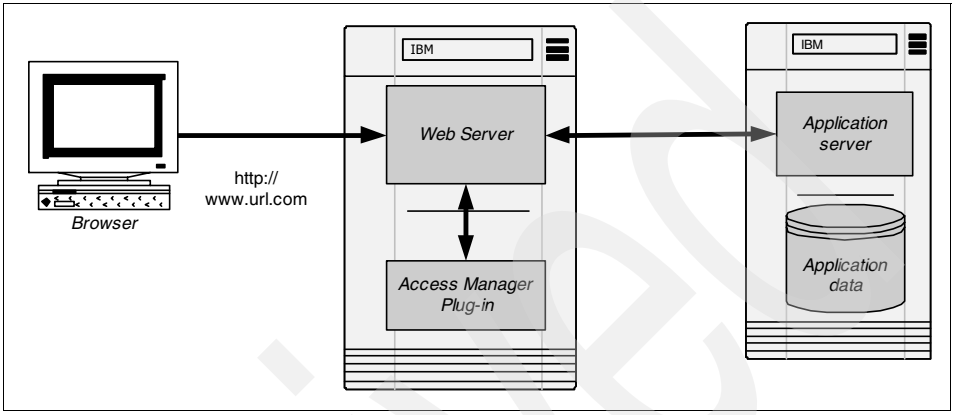


Figure 6-14 Plug-in logical architecture

6.7 Component configuration and placement

Obviously, it is possible to deploy Access Manager components within a single network. While this kind of architecture may be reasonable for a lab or development environment, it generally is not for a production setting. Most Access Manager deployments must fit within the context of network security requirements.

In this section, we discuss the ways various Access Manager components relate to the network configuration and provide recommendations for their distribution in a typical architecture.

6.7.1 Network zones

In Chapter 2, “Common security architecture and network models” on page 19, we discussed network zones and their relationship to security. Here, we discuss these zones in the specific context of Access Manager architecture.

We have to consider four types of network zones in our discussion of Access Manager component placement:

- Uncontrolled (the Internet)
- Controlled (an Internet-facing DMZ and the intranet)
- Restricted (a production network)
- Secured (a management network)

Because we will not place any components in an uncontrolled zone, we look at the remaining three zones.

Internet DMZ (controlled zone)

The Internet DMZ is 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 typically is bounded by two firewalls, there is an opportunity to control traffic at multiple levels:

- Network: IP addresses, NATs, and so on
- Protocol: HTTP(S), FTP, SMTP, and so on
- Application: Application proxy, terminal services, and so on

WebSEAL or a Web server plug-in (with no data content) fits nicely into such a zone, and in conjunction with the available network traffic controls provided by the bounding firewalls, it provides the ability to deploy a highly secure Web presence without directly exposing components that may be subject to attack by network clients.

Production network (restricted zone)

One or more network zones may be designated as *restricted*, meaning that 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 and outgoing traffic may be filtered as appropriate.

These zones typically contain replicated information of user registries and access control information in order to provide this information as close to the decision-seeking applications as possible.

Management network (secured zone)

One or more network zones may be designated as a *secured* zone. Access is available only 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*.

These zones typically contain *back-end* Access Manager components that do not directly interact with users.

Other networks

Keep in mind that the network examples we are using do not necessarily include all possible situations. There are organizations that extensively segment functions into various networks. Some do not consider the intranet a controlled zone and treat it much like the Internet, placing a DMZ buffer between it and critical systems infrastructure contained in other zones. However, in general, the principles discussed here may be easily translated into appropriate architectures for such environments.

Placement of various Access Manager components within network zones is both a reflection of the security requirements in play and a choice based on existing or planned network infrastructure and levels of trust among the computing components within the organization. Requirement issues often may be complex, especially with regard to the specific behavior of certain applications, but determination of an Access Manager architecture that appropriately places key components usually is not difficult. With some knowledge of the organization's network environment and its security policies, reasonable component placements are usually easily identifiable.

Figure 6-15 summarizes the general Access Manager component type relationships to the network zones discussed previously.

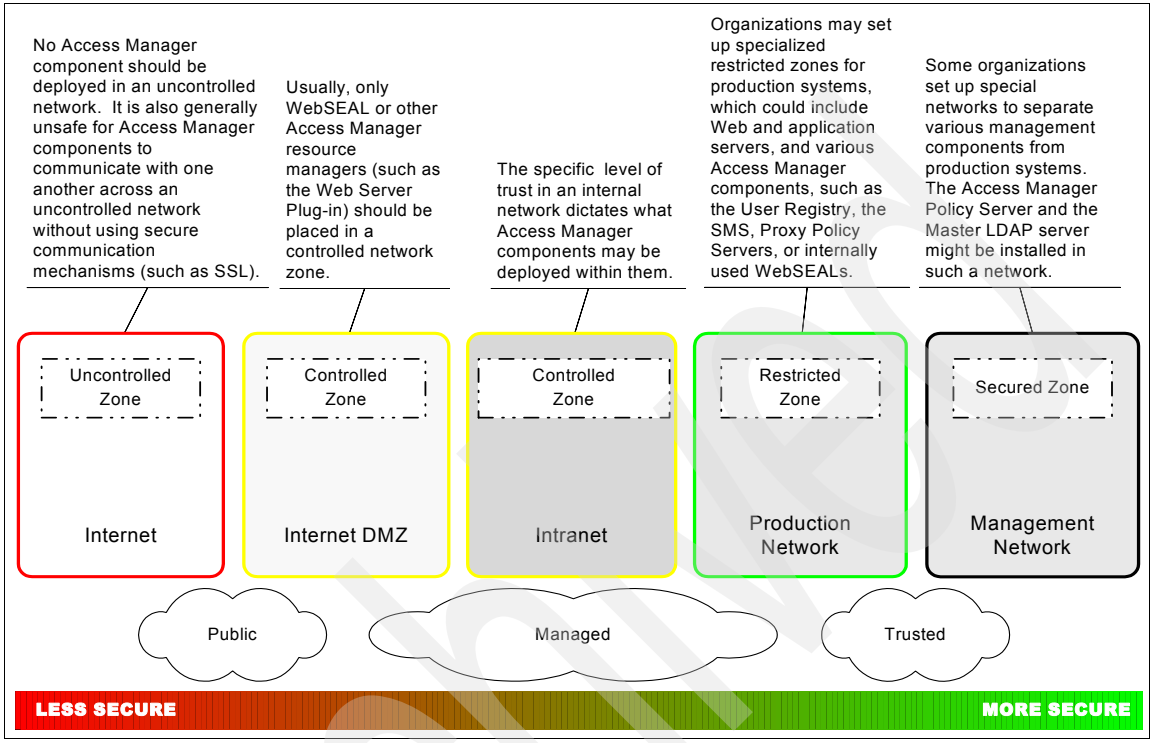


Figure 6-15 Network zones

6.7.2 Secure communication issues

All communication among Access Manager components is configurable to be secured using SSL, which addresses the issues of privacy and integrity of communication among components. It does not deal with other types of security exposures that are inherent in the physical placement of components within the network infrastructure, such as system hardening and application security.

The choice to use SSL among certain components should be primarily based on the trust relationships that exist *within* and *between* the network zones in which they operate. While trust may influence the placement of various Access Manager components within different network zones, the use of SSL itself does not govern such placements.

6.7.3 Specific Access Manager component placement guidelines

Now that we have discussed the basic issues involved in component placement, we can go into greater detail regarding specific components typically found in a Access Manager Web-based architecture.

Policy Server

The Access Manager Policy Server should always be placed in a secured (or at least a restricted) zone. Figure 6-16 summarizes the guidelines for placement.

In the case of using the Policy Proxy Server it should be placed in a more trusted zone adjacent to the location of the Access Manager applications. For example, in Figure 6-16 assume that a WebSEAL is located in the Internet DMZ and the Policy Server is in the management network. It might be a good idea to use the Policy Proxy Server within the intranet so that no direct connections are allowed from the Internet DMZ to the management network.

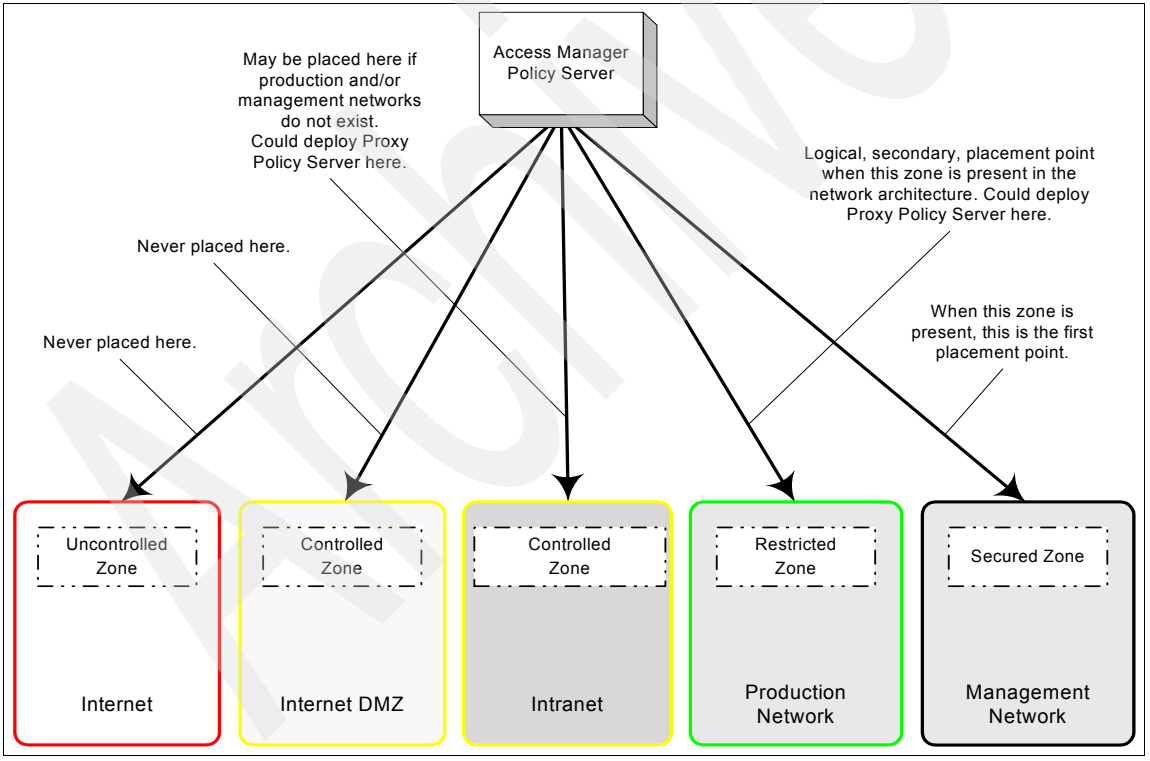


Figure 6-16 Policy Server placement guidelines

Note: Using the Policy Proxy Server as described is based on the same principle as the use of WebSEAL in the DMZ for inbound Internet connections. This is called defense in depth. In the Internet instance, WebSEAL acts as a reverse proxy (buffer) between the less-trusted Internet and the more-trusted production network. The Policy Proxy Server acts as a buffer between the less-trusted DMZ and the more-trusted management network.

User Registry

As previously discussed, WebSEAL interacts with the Access Manager User Registry to perform some of its functions. This means that the registry must be accessible to WebSEAL. However, it probably should not be accessible to general users, especially from the Internet.

The registry should be in a restricted zone to which access may be strictly controlled. Firewall configurations should disallow any possibility of access to the User Registry from the uncontrolled zones such as the Internet (for example, port 389 access might be disallowed by an Internet-facing firewall, and outgoing port 389 accesses only allowed to pass from the Internet DMZ to another zone if initiated by a WebSEAL server).

Figure 6-17 summarizes network zone placement guidelines for the User Registry.

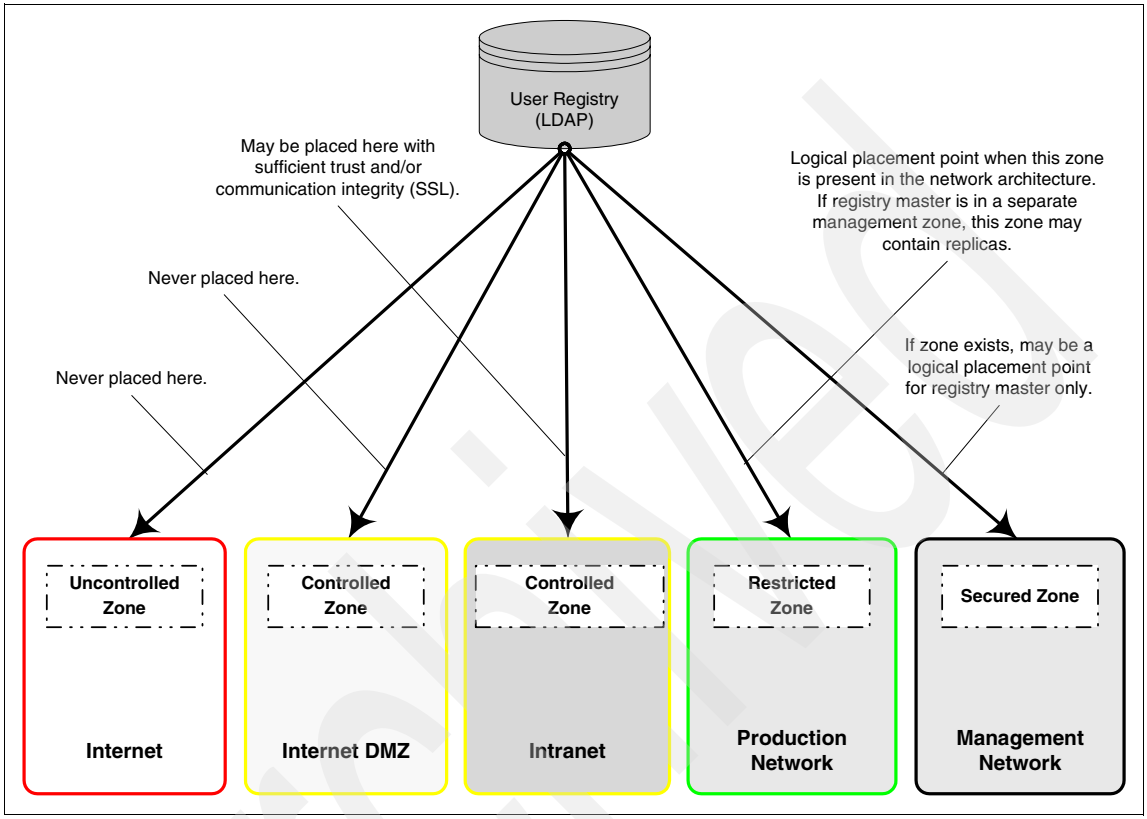


Figure 6-17 User Registry placement guidelines

Figure 6-18 shows an example of User Registry placement using network filtering rules to limit access.

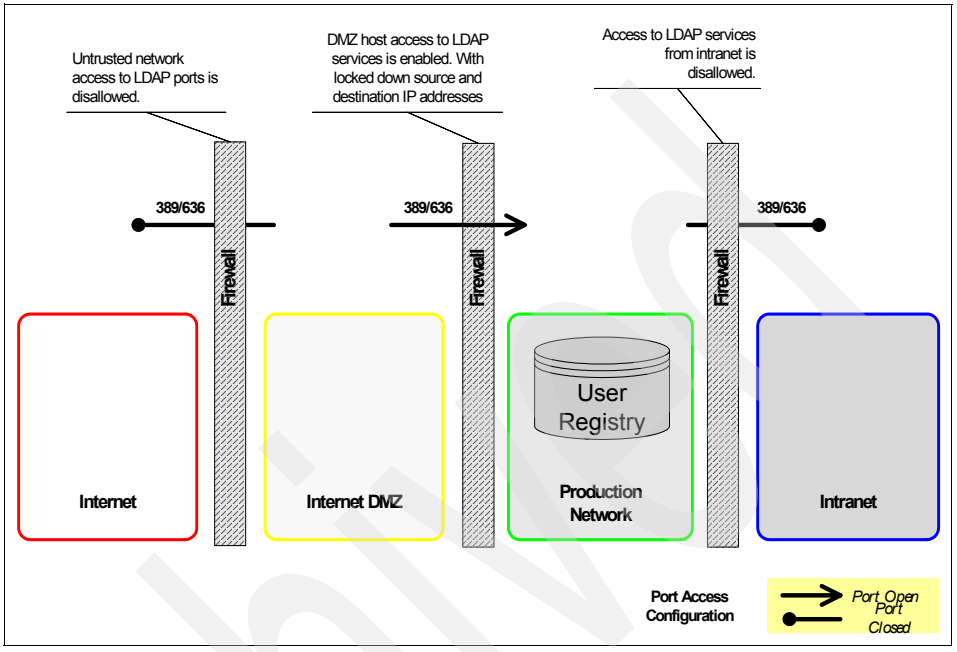


Figure 6-18 Restricting network access to User Registry

Additionally, it may make sense to separate the *read* functions of the registry that are needed by WebSEAL from the *write* functions that are required by Access Manager management components. This can be done by creating a registry replica used for *read-only* access (such as authentication) and leaving the registry master only for making updates. If there is a special *management zone* into which all management components must be placed, such a configuration may be appropriate. Figure 6-19 on page 229 shows an example of this.

Tip: When deploying WebSEAL or Web server plug-in nodes, the user registry should be as near as possible to these to provide maximum performance and authentication speed. This implies that, generally, a replica of the user registry will be required in the more-trusted zone adjacent to that where the WebSEAL or Web server plug-in nodes are deployed.

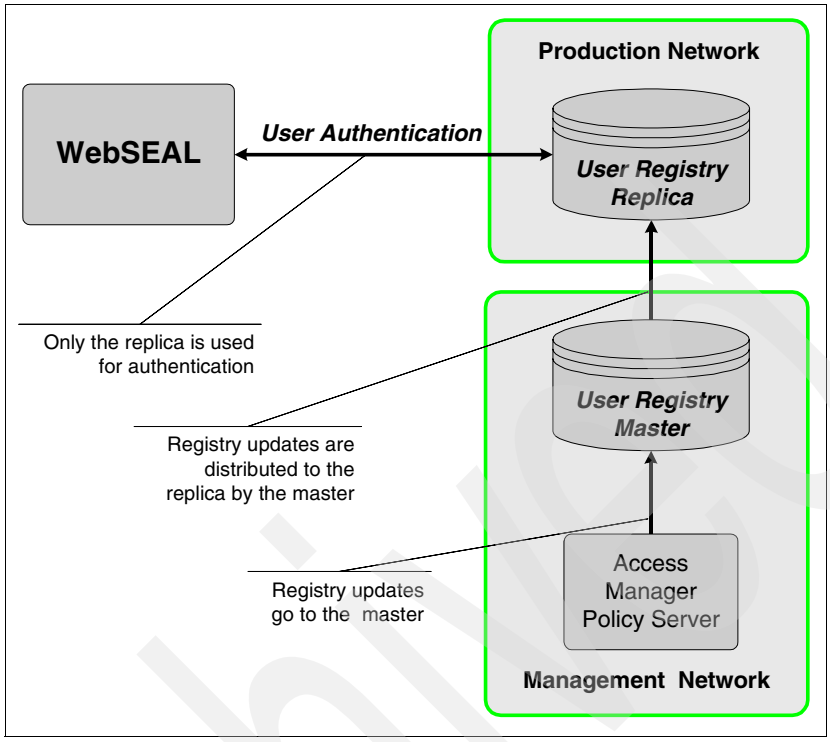


Figure 6-19 Separating User Registry read and write functions

Web Portal Manager

The Web Portal Manager should always be placed in a restricted zone (or at least a trusted zone). If a separate Management DMZ is used, there may be issues in how best to structure the configuration of the Web Portal Manager in such an environment.

Because the Web Portal Manager's functions are accessed via HTTP/HTTPS, access to it can be configured via a WebSEAL junction. If this is done, special consideration should be given to its placement and how access should be controlled.

Figure 6-20 summarizes placement guidelines for the Web Portal Manager.

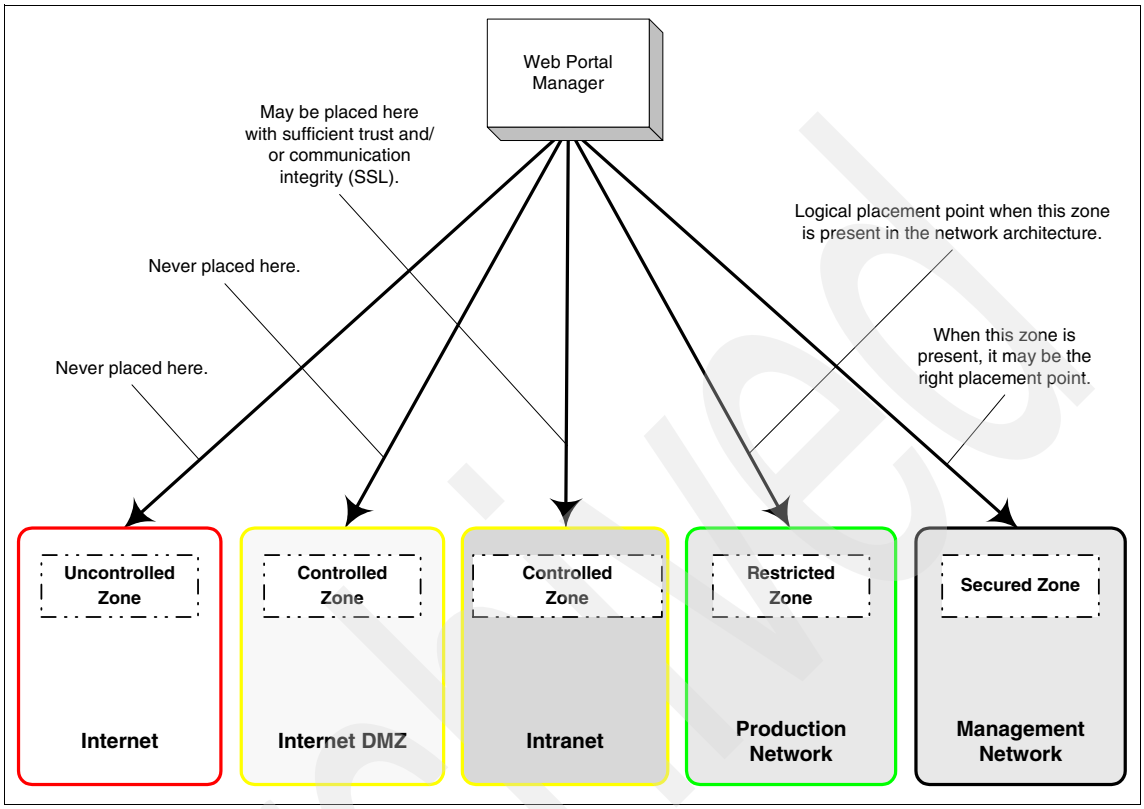


Figure 6-20 Web Portal Manager placement guidelines

WebSEAL

WebSEAL should always be the sole HTTP/HTTPS contact point for a Web server from an Internet client. When using WebSEAL in an intranet setting, this is usually desirable as well.

Internet

Based on our discussion so far, it should be clear that WebSEAL servers accessible via the Internet should be placed in a DMZ. WebSEAL in such a setting should generally be in a network zone separate from those that contain other Access Manager components upon which it relies, and from the Web servers to which it is junctioned.

In general, the DMZ network boundaries are best secured through firewalls, and appropriate traffic filters are used for strict control of the flows into and among

components. In this case, the Internet-facing firewall should be configured to make ports 80/443 accessible only through WebSEAL, as shown in Figure 6-21.

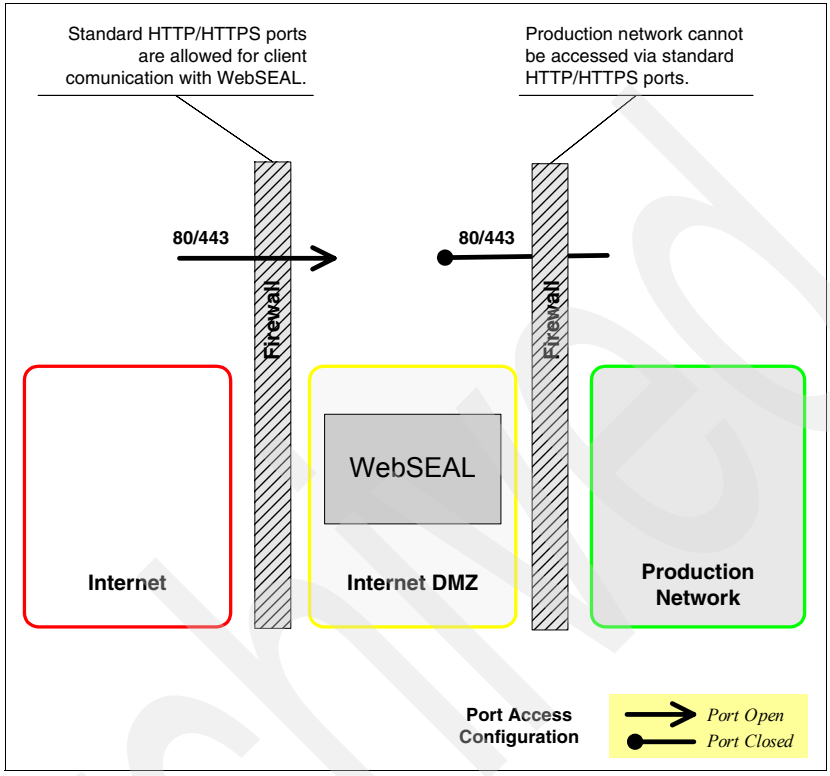


Figure 6-21 Restricting HTTP/HTTPS network traffic paths

This approach has several advantages:

- It focuses all Web traffic through a single path.
- Secured Web content is not directly accessible.
- Compromise of the Internet-facing firewall results in limited security exposure.

This illustrates a key WebSEAL strength: As a reverse proxy, it provides security capabilities that cannot be supported by any other approaches, such as plug-ins.

WebSEAL minimizes the numbers of hosts that must be placed in an Internet DMZ. In addition to the security benefits for businesses that utilize hosting services to support their DMZs, this may enable them to reduce costs by moving substantial amounts of Web infrastructure back into their internal networks, leaving WebSEAL hosts as the key component in their hosted environments.

Intranet user access via WebSEAL

WebSEAL may also be used to serve Web content to internal clients. Certain issues must be addressed when using it in this manner.

It may seem reasonable to simply force internal clients to use the same WebSEAL hosts that are serving Internet clients. However, such an approach may not be best because a security compromise of the Internet DMZ could create direct attack paths to internal clients.

An alternative approach is to dedicate a separate WebSEAL server for internal uses and place it in an appropriate internal network zone. Depending on the level of trust and other configuration factors, the following choices exist for placement of an internal WebSEAL server:

- Place the WebSEAL server in the same network zone as other Access Manager components.
- Place the WebSEAL server in an internal DMZ that is separated from other Access Manager components (essentially, mirror the Internet DMZ scenario internally).

Given a sufficient level of trust internally, it may be reasonable to choose the first approach and put the internal WebSEAL in the same zone as other components. This approach is often chosen when architecting WebSEAL solutions for internal user access.

For environments in which the internal trust is insufficient to justify placing WebSEAL into a common zone with other components, the second approach may be more appropriate.

WebSEAL placement summary

Figure 6-22 summarizes the guidelines for WebSEAL placement.

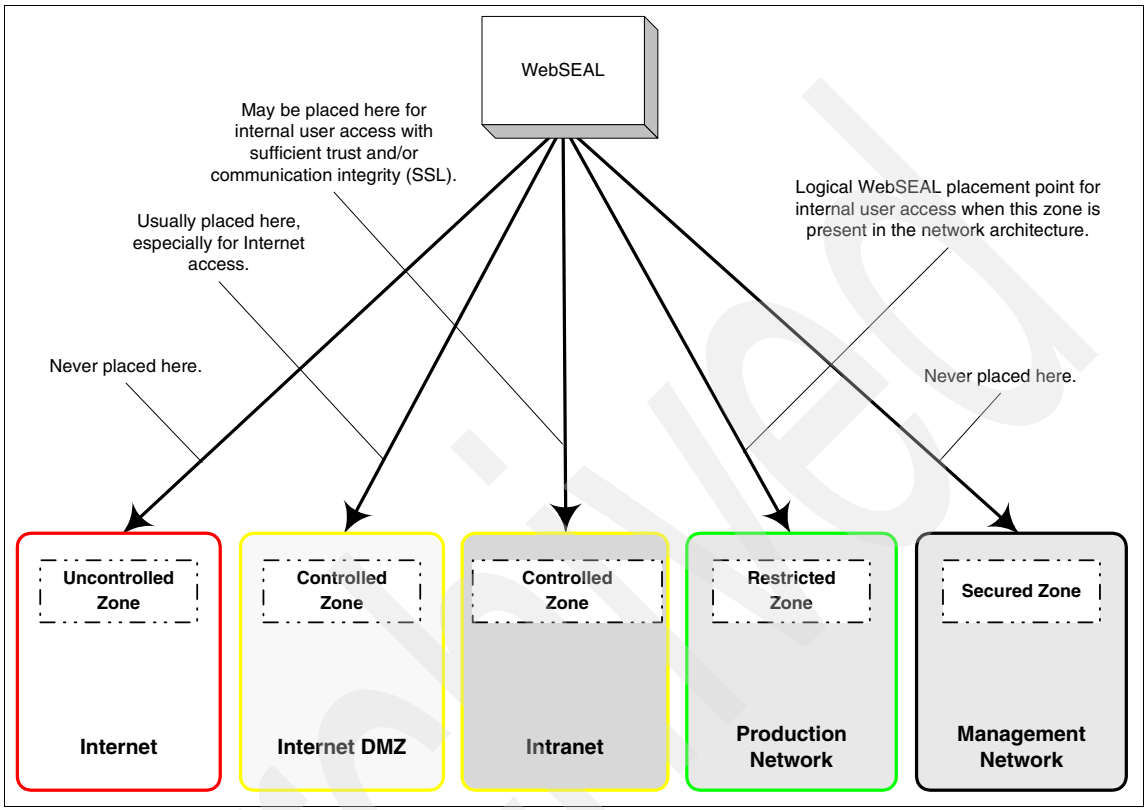


Figure 6-22 WebSEAL placement guidelines

Junctioned Web servers

In a WebSEAL configuration, it is recommended that junctioned Web servers not reside in an Internet DMZ. WebSEAL does not restrict Web server placement in any way, but the further away one can move critical resources from uncontrolled zones, the better.

Ideally, Web servers should be in a special, restricted zone, but could also be placed in a more open yet trusted network zone if appropriate configuration steps are taken, such as utilizing SSL for communication with WebSEAL and configuring the Web server so that it will accept only connections from a WebSEAL host). Figure 6-23 summarizes the zone placement guidelines for Web servers that are junctioned via WebSEAL.

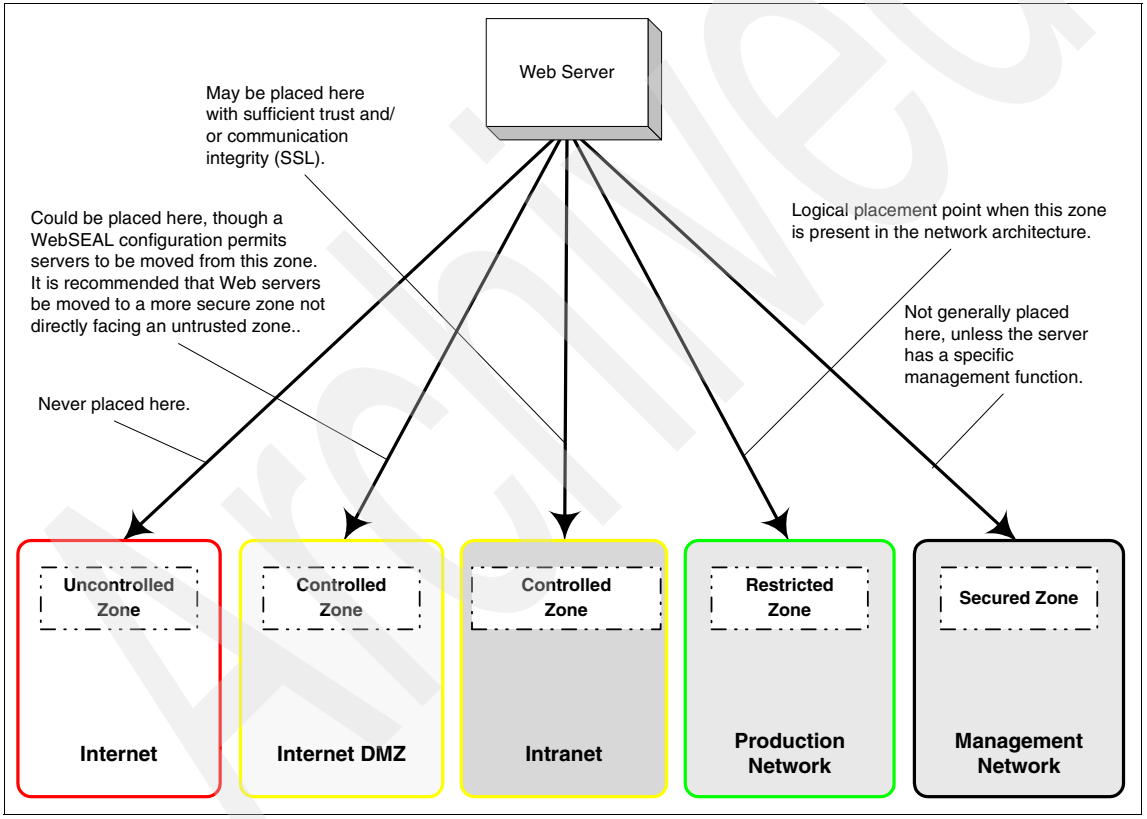


Figure 6-23 Web server placement guidelines

It may be a good idea to configure junctioned Web servers to use ports other than 80/443 (for example, 81/1443). This enables the Internet DMZ firewall configuration to be structured such that port 80/443 access can only be made to the Internet DMZ, and the internal-facing firewall can be configured to disallow ports 80/443 and only allow these alternate ports into the restricted/trusted zone. Such a configuration is exemplified in Figure 6-24.

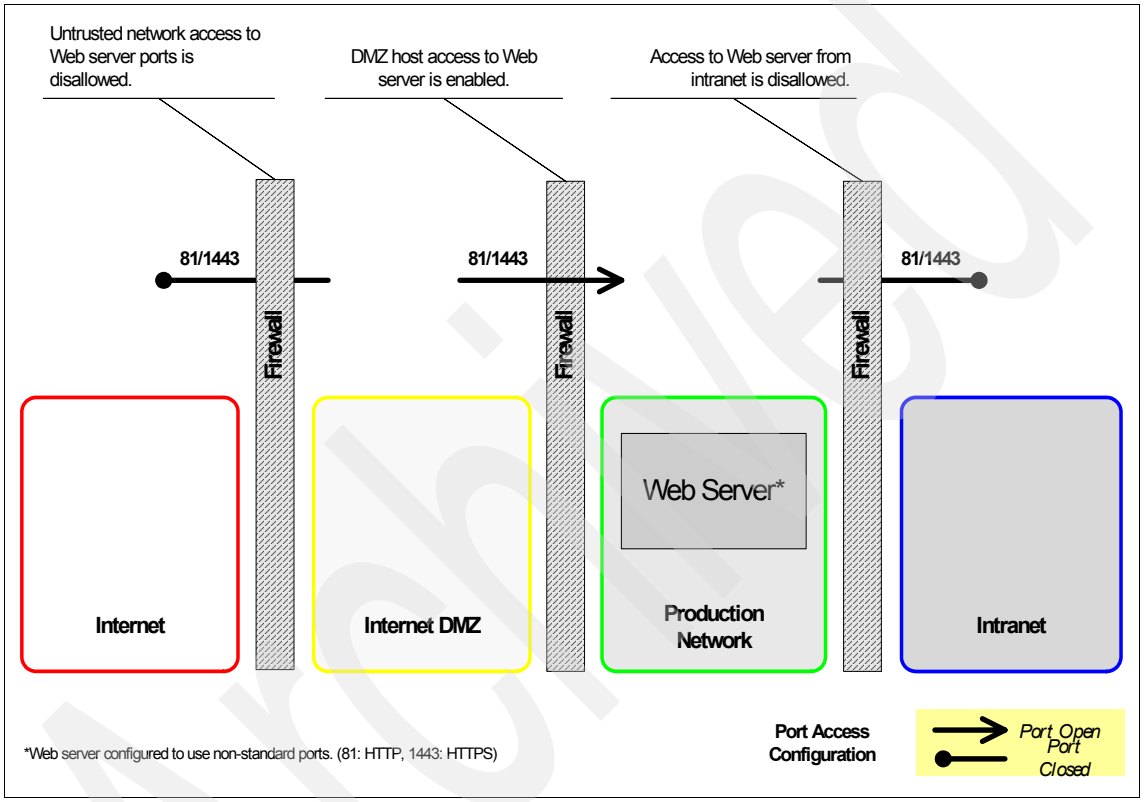


Figure 6-24 Limiting network access to Web servers

Web server plug-in

Based on the previous discussion it is easy to understand how the Web server plug-in fits into the current architectural discussion. When utilizing a WebSEAL architecture the Web servers normally reside within either the production network or the intranet (depending on the overall environment and security requirements). Further, from a security standpoint, it is possible to run the Web servers on the same physical nodes as the application servers.

When WebSEAL is not used, the Web servers must be moved into the DMZ to provide the first point of contact for client connections. This also implies that it will not be possible to run the application servers and Web servers on the same node, as it is not advisable to run application servers in the DMZ. Figure 6-25 shows how this architecture could be implemented.

The architectural discussions in previous sections about the core Tivoli Access Manager components (such as the user registry, ACL database, and Policy Server) are still valid when using the Tivoli Access Manager Web server plug-in.

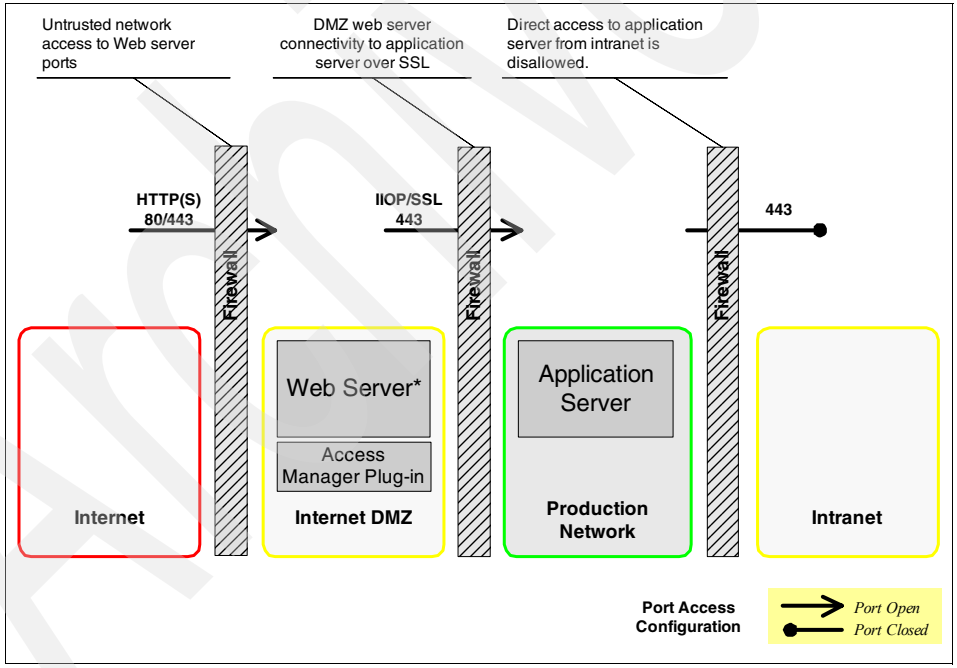


Figure 6-25 Plug-in architecture

Note that there is no layer of protection for the Web servers. At an application level, the authentication and authorization capabilities are provided by Tivoli Access Manager with all of the added advantages of centralized management, common authorization services, and audit. However, at a system and network level, Internet users have direct access to the Web servers. This exposes the Web servers to all of the inherent threats that originate from the Internet.

Important: The previous sections are intended to help the reader understand the architectural difference between the use of WebSEAL and Tivoli Access Manager Web Plug-ins. Careful consideration must be given to which approach an organization should adopt.

Issues such as risk appetite, infrastructure and operational cost, security policy, and business drivers and strategy must be addressed and balanced by a qualified architect to enable the appropriate organizing decision to be made.

Putting it all together

Now that we have discussed the placement of the various components in a WebSEAL configuration, we put it all together in a typical architecture. Assume that the following network zones exist:

- An uncontrolled Internet zone
- A controlled Internet DMZ zone
- A restricted Production Network zone

Without discussing the specific requirements of the organization, let us assume a basic WebSEAL configuration for both Internet and internal user access. One possible architecture could be as depicted in Figure 6-26 on page 238.

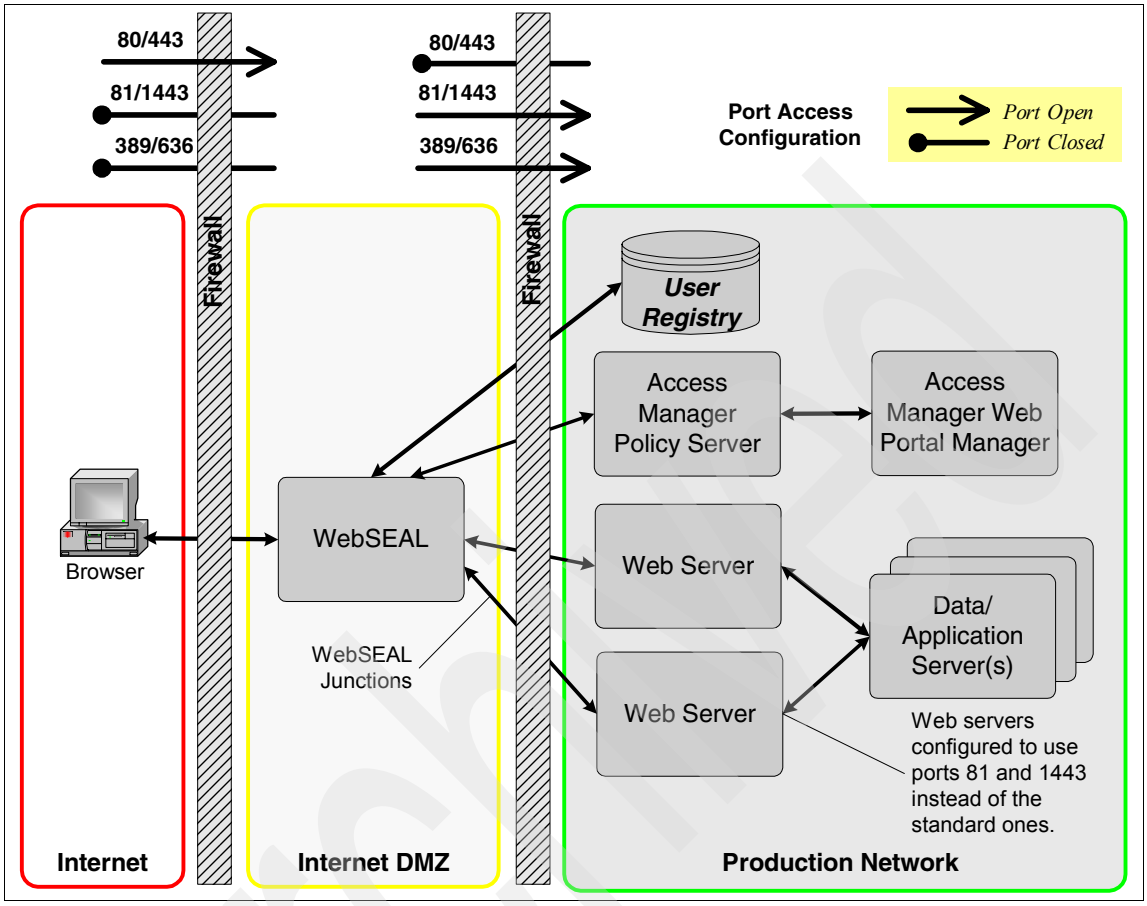


Figure 6-26 A sample Access Manager WebSEAL architecture

It should be clear that by simply following the guidelines, many Access Manager WebSEAL architectures are relatively straightforward. The real complexities often come into play when addressing things other than the overall architecture itself, which are normal issues involved in enterprise systems deployment. This includes such things as configuration, deployment plans, capacity requirements, operational policies and procedures, and specific application integration issues.

6.7.4 Summarizing Access Manager component placement issues

In the previous discussion, it must be emphasized that, to a large extent, the placement of Access Manager components represents a set of choices. Nothing in Access Manager itself dictates what kind of network configuration is required. The component placement guidelines are actually related more to overall

security principles than to any particular Access Manager need. In fact, in a WebSEAL deployment such as we have discussed in this chapter, Access Manager actually offers greater component placement flexibility than many other approaches to Web security.

This said, keep in mind that you cannot simply separate network configuration issues from Access Manager. While Access Manager components perform their duties extremely well, good sense dictates that they must operate in an environment that prevents them from being bypassed and protects them from undue exposure to other forms of attack. With *any* security solution, not just Access Manager, this must be kept in mind.

6.8 Physical architecture considerations

In our discussion of WebSEAL architecture so far, we have focused primarily on the logical relationships among software components and not necessarily on specific system configurations upon which they are installed.

6.8.1 Access Manager components

It should be clear from our earlier discussion that, at least for Internet scenarios, WebSEAL should reside on a separate host from other Access Manager components.

However, where other (back-end) components should go is not as clear. There are no “rules” regarding this. Where these components should be placed is dependent on a number of factors, including:

- The specific network configuration within which Access Manager is installed
- The capacity and capability of the host systems on which these components are installed
- The amount of flexibility required for future expansion of the security infrastructure
- Specific security or operational policies that may dictate certain Access Manager configurations

It is possible to place all required back-end Access Manager components on a single host system. However, other than in a very simple WebSEAL deployment or a lab setting, this may not be the best approach. For example, a common way to break things out would be to place the management functions on one host and the User Registry on another. Figure 6-27 shows a physical system layout mapping of the example architecture shown previously in Figure 6-26 on page 238. Keep in mind that this is simply an example and it does not represent the only way in which components may be combined on host systems.

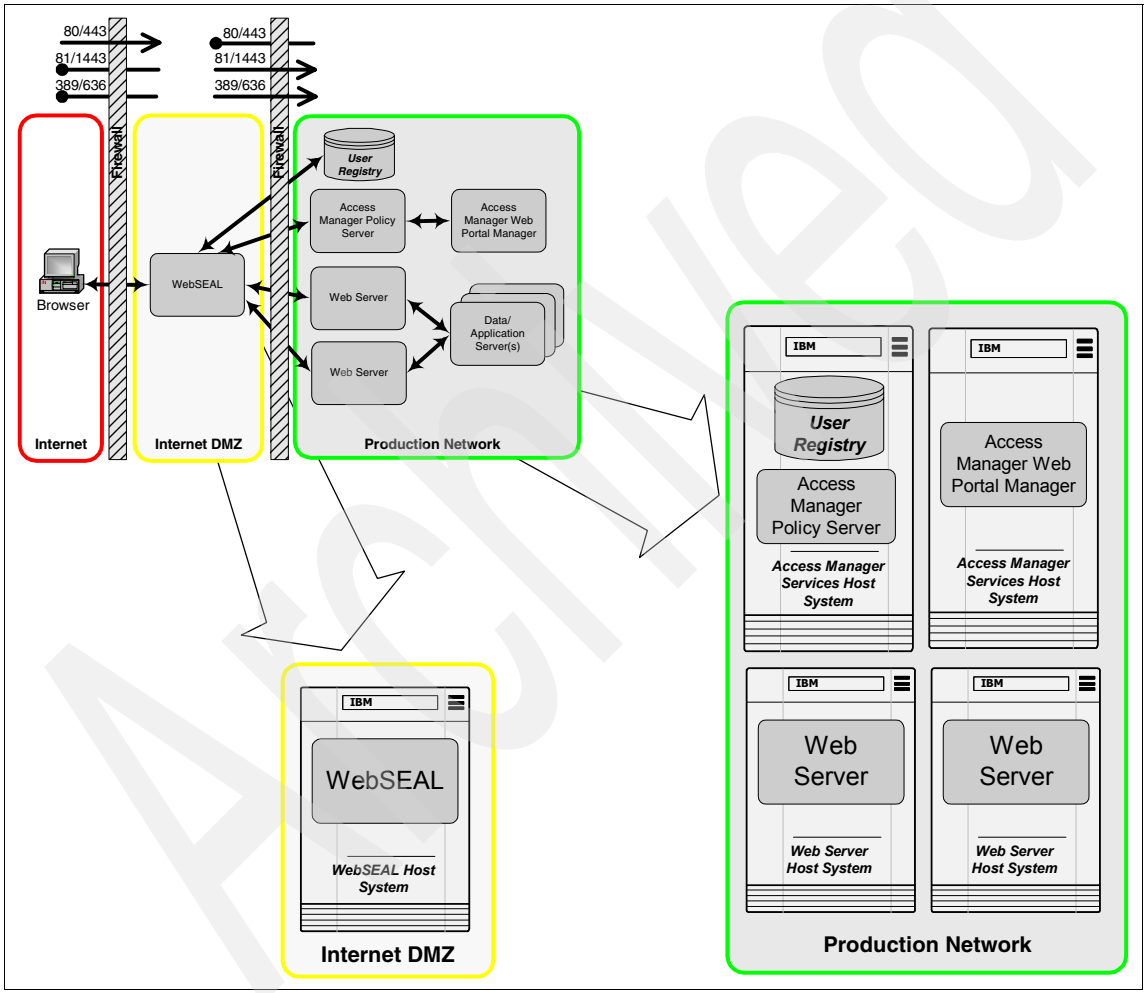


Figure 6-27 A sample physical component layout

6.8.2 Other infrastructure components

In addition to Access Manager components themselves, other components are a natural part of the infrastructure in most typical environments, including:

- Domain Name Service (DNS) or other, similar naming services
- Time services, such as Network Time Protocol (NTP)
- Host configuration services, such as Dynamic Host Configuration Protocol (DHCP)
- Mail transport agents (MTAs), such as sendmail
- File transfer services, such as FTP

Domain Name Service

In general, Access Manager components themselves should avoid the use of naming services for address resolution. It is usually best to directly configure host addresses locally, both for availability and security reasons.

In cases where access to a name service is needed by a Access Manager host, consideration should be given to installing a DNS secondary on the host itself or in close proximity to the host in an appropriately protected network zone. In no case should the security infrastructure share DNS services with the general user community, either internal or external.

Another note regarding the use of DNS in an Internet WebSEAL setting. It is recommended that a split-level DNS configuration or other approach be employed to ensure that external clients have no IP address resolution visibility beyond the WebSEAL hosts themselves.

Time services

Time services are required when more advanced configurations of Access Manager for e-business are used such as the Session Management Server. If a robust solution is not implemented, it is still a good idea, if for no other reason than to assure that audit logs contain consistent time stamps. Network Time Protocol (NTP) is the recommended choice for time synchronization, and an appropriate implementation should be available on all platforms on which Access Manager runs.

Host configuration services

Host configuration services, such as DHCP, should never be used by any host running Access Manager components. IP addresses should be statically configured. It is also recommended that DHCP services not be provided by hosts that are running Access Manager components.

Mail transport agents

Mail transport agents, such as sendmail, are often present within the network infrastructure to route mail both internally and externally. Such mail gateways should not be configured on Access Manager hosts, as their use may affect system performance characteristics substantially and diminish performance predictability.

Additionally, a WebSEAL host, especially one that is accessible via the Internet, should not respond to SMTP (port 25) connection requests.

File transfer services

File transfer services, such as anonymous FTP, are often present within the network infrastructure to support access to program archives or other information. It is recommended that such services should not be configured on Access Manager hosts, as their use may substantially affect the performance characteristics of the system and diminish performance predictability.

Additionally, a WebSEAL host, especially one that is accessible via the Internet, should not respond to FTP (port 20) connection requests.

6.8.3 General host hardening considerations

In addition to the recommendations given so far, it may make sense to harden certain hosts that participate in an Access Manager configuration. This may be especially true for Internet-facing WebSEAL hosts.

While the specifics of hardening an operating system are beyond the scope of this book, the following items are representative of the types of issues addressed:

- The number of incoming paths through which it may be accessed is minimized (for example, turning off certain network services that are not necessary for system operation).
- The number of outgoing paths from the system to other hosts is minimized (for example, limiting the system's knowledge of other hosts to those absolutely necessary for proper operation).
- Appropriate system auditing functions are enabled to assure traceability of accesses.
- The set of users that may access the system is minimized to a level that is necessary for system operation, and clear roles and responsibilities are defined for those users (and, where possible, enforced).

Additionally, certain network firewall configurations may be employed to enforce the restrictions of a hardened environment.

6.9 Access Manager: Part of overall security solution

It would be a mistake to assume that deployment of WebSEAL or the Web Plug-in alone is sufficient to fully address all security requirements. Access Manager provides key functionality, which is essential for Web security, but it cannot cover all contengencies. As should be evident from the discussion of other topics in this book, other security considerations should be addressed in conjunction with Tivoli Access Manager.

We have not discussed other security components that may work in conjunction with Tivoli Access Manager and other Access Manager components to address broader security concerns. In particular, Identity Manager and Security Operations Manager, which are discussed in Part 3, “Managing identities and credentials” on page 507 and Part 5, “Managing security audit and compliance” on page 843, provide functionality complementary to Access Manager and can be of substantial value as components of an overall security solution.

7

A basic WebSEAL scenario

Our earlier discussion of Access Manager has been helpful in describing the basic elements of architecture for deployment. At this point, we apply those guidelines to a simple Web scenario for a fictional organization with a typical set of requirements.

In our discussion, we deliberately avoid certain issues, including availability considerations and specific issues relating to application integration. These areas are discussed in later chapters.

Also, while host machine configuration and capacity is touched upon in this chapter, we deliberately avoid providing much in the way of specifics. This is because without appropriate capacity planning activities, which consider simulated/real loads of the actual application, accurate determinations can be difficult to make.

7.1 Company profile

Stocks-4u.com is a wholly owned United States subsidiary of a major brokerage company, Medvin, Lasser & Jenkins (ML&J). Until now, ML&J's online presence has been limited, consisting mainly of informational Web content. Online trading has not been a priority. The clientele traditionally has been major accounts with assets greater than US$5 million, and transactions are almost exclusively done via direct contact with a broker. While the company, a privately held corporation, has maintained solid profitability over the past several years, largely due to a stable client base, the company's growth has stagnated, remaining at approximately the same revenue levels since 1995.

Market trends have forced a rethinking of ML&J's approach to business. The individual investor community has increased substantially in recent years, and the company has not shared in that growth. Consequently, the company's market share has eroded. Also, the rise of online trading has begun to affect a portion of ML&J's client base. In the past year, there has been a net outflow of investment funds cutting across approximately 10% of all client accounts. Research has shown that 95% of these outflows are being redirected to online brokerages. This trend, if it continues, threatens to affect the long-term viability of the business.

An online component to complement ML&J operations has been judged a necessity. Stocks-4u.com was started with assets recently acquired from a failed Internet startup. Additional capital has been provided to fund completion of the company, which recently began full production operation ramp-up. Stocks-4u.com services the online trading requirements of ML&J's current clients while focusing on developing additional clients who are primarily online traders with trading capital in excess of $250,000.

Note: As of April 2007, our fictitious domain name Stocks-4u.com was not reserved by anyone.

7.2 Technology background

Stocks-4u.com has been deployed as a Web-based online trading system with capabilities similar to those found at other online trading sites. This software consists of a number of underlying applications, all of which perform functions based on the each user's privileges. For example, only users who have paid for Level II quotes may access that application.

In concert with the ongoing application development activities, the company has been examining alternatives for providing secure access to their Web site. Originally, a master application was developed that provided a single access

point for providing user authentication and authorization utilizing the underlying capabilities of the operating system.

Following initial deployment, additional requirements became apparent. It became clear that the level of effort required to fully address all functional requirements was cost-prohibitive. The tie-in to the operating system security mechanisms began to limit certain deployment options. The CIO felt that this approach was locking them in architecturally to an in-house solution that would require long-term sustaining and support services. After examining marketplace alternatives in a proof-of-concept (POC) setting, a decision was made to deploy an Access Manager security capability, leading with Web security.

The company wants to transition its user base from the in-house Web security system to a WebSEAL-based system over the next few months. Initially, they want to deploy adequate capacity to address their anticipated loading over six months, and then incrementally add more as needed.

7.3 IT infrastructure

The Stocks-4u.com concerns for becoming an integral part of the ML & J IT infrastructure fall into three major categories:

- Data centers
- Network
- Operational plans

7.3.1 Data centers

Stocks-4u.com has two major data centers. One is located in San Diego, California, and the other is in Savannah, Georgia. At this time, all Internet application access and key internal application access is provided through the San Diego center, in which the company's IT Operations (OPS) group is based. The Savannah center currently supports a few other internally used applications and houses the company's IT Architecture, Development, and Deployment Support (ADDS) business unit.

Stocks-4u.com considered hosting its Web servers through a third-party provider, but it was decided that all subsidiaries would deploy its servers in-house. However, they have not ruled out migrating certain Web operations to a hosting provider in the future. This could bring additional data centers into play.

7.3.2 Network

The data centers are connected by redundant T3 (45mbps) access. At this time, Internet connectivity is provided through the San Diego center, with multiple T3 lines from three different providers. The diagram depicted in Figure 7-1 shows the national Stocks-4u.com network.

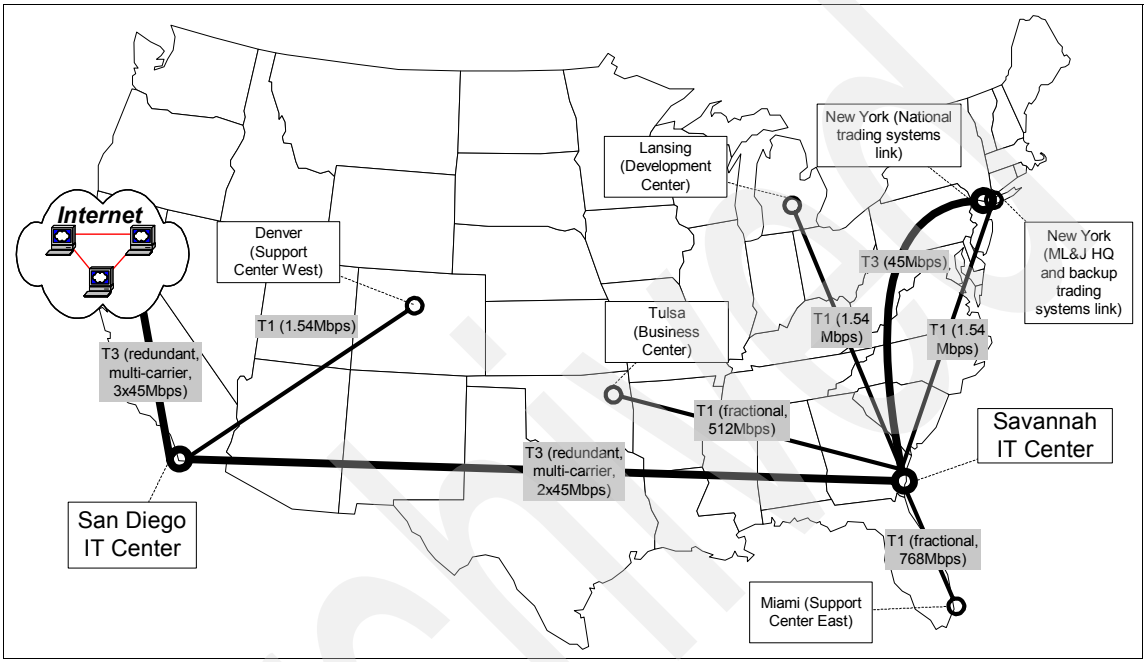


Figure 7-1 Stocks-4u.com data network

Within the San Diego center, all Internet access is channeled through Web servers residing in a demilitarized zone (DMZ). These Web servers provide front-end application logic, including presentation services. Back-end application logic is hosted on systems residing behind the DMZ in an internal production network.

The Savannah center has no direct Internet access. It has a production network for internal application systems.

In addition to the specific network capabilities at each of the sites, there is also a general company intranet shared across all corporate locations. This network is not considered secure and is not authorized for hosting production systems.

7.3.3 Operational plans

Early plans are in the development stage for future expansion of Internet operations into the Savannah center to provide for a redundant access capability with load-balancing for customers on the U.S. East and West coasts. At this time, there is no requirement to actually support this. However, the Stocks-4u.com chief architect wants to be certain that the security solution they deploy is capable of meeting such a requirement. During the Access Manager proof-of-concept, it was determined that this should not be a problem.

7.4 Business requirements

The CIO has provided input about the business drivers for the targeted solution:

- Provide an enabler for consistent application of security policy across the business. The business cannot afford to create multiple, competing security infrastructures.
- Assure client confidence by offering a flexible yet perceptively secure solution. It is essential that the security system not get in the way, while at the same time protecting client information and assuring that financial transactions are conducted securely.
- Competitively position the business to react quickly in deploying secure premium services and content. Quickly deploying value-add capabilities is important to gaining and maintaining market share.

  Allow for the integration of special premium application capabilities to ML&J's "Select" clients. The firm is very focused on maintaining their existing high-income client base by providing them with special capabilities that are not available through any other online service. For example, additional bond management capabilities within the portfolio management application are being developed specifically for these clients.
- Provide for expansion of services with minimal incremental investment. It is essential that, once in place, the security solution grow with the company. It is unacceptable to require extensive and continuing re-engineering efforts for the security infrastructure as the company expands its operations.
- Meet applicable U.S. Securities and Exchange Commission (SEC) requirements. There are certain legal requirements for assurance that client assets and transactions are handled properly. The security infrastructure should be supportive of these requirements.

7.5 Security design objectives

Based on initial discussions and a security workshop, it has been determined that the following key technical requirements exist:

- Provide a single sign-on capability for all Web-based applications. A user should only have to log in one time to one entity to obtain access to all authorized applications and content that may reside on various servers.
- Remove the need for application developers to authenticate users. The company does not want to invest in developing any authentication capabilities within its new applications.
- Provide a cross-platform security solution. Previous experience with the in-house security application clarified the need to maintain operating system independence for Web-based application security.
- Provide the ability to control access to Web applications and content, which may be hosted through multiple Web servers, at the URL level.
- Provide the ability to control access to applications that have existing URLs without having to modify the application or the URL. One main application is ticker.stocks-4u.com.
- Provide the ability to make fine-grained authorization decisions within applications. While this is not an immediate deployment requirement, the solution must allow for this capability to be added.
- Support browser-based access to applications from both employees and customers. From their desks, internal users may access both Internet-hosted applications and internal applications. At this time, there is no requirement for employees to have access to internal applications from the Internet.
- For the first six months following deployment, load requirements are for up to 40,000 Internet users, with an annual growth rate of 50% over the next five years. In five years, the online client base is expected to exceed 300,000 users. Approximately 25% of all clients are expected to conduct at least one transaction on any given day.
- The internal employee user base is currently around 250 and is expected to grow to approximately 1000 during the next five years. Approximately 80% of employees are expected to conduct at least 10 transactions on any given day.

7.6 Requirements analysis

The requirements for this access control subsystem are typical of those found in many Web application environments. Also, Stocks-4u.com's experience with home-grown security is not unique. With today's Web-centric application focus,

many organizations approach the security issue from that perspective, yet they often utilize existing host-based security systems that prove inadequate for addressing key requirements. The fact is that, while some host-based security capabilities are extensive, they are tied to a specific platform. This is inconsistent with the reality of today's Web-based applications. These applications often run on several different machines on several different platforms and on various Web server implementations.

An Access Manager WebSEAL capability is an obvious fit for Stocks-4u.com's current needs. In fact, most Access Manager deployments start with a Web focus. However, there are clear requirement statements that discuss future infrastructure expansion, and the same Access Manager environment that supports WebSEAL will also be capable of addressing those needs.

For example, it is clear that the company has a future need to support a tighter application-level integration with security, using Authorization Application Programming Interface (aznAPI) or JAVA2 security-based functionality to allow very detailed authorization for application components. The inherent architecture of Access Manager enables these requirements to be met easily.

In this example, we address the immediate requirements of Stock4.com with a WebSEAL solution. However, in a later chapter of this book, we may introduce additional requirements or revisit some of the remaining issues to illustrate how they may be addressed as the company expands its use of Access Manager.

To summarize the requirements discussion above, we know the following:

- We need to have a WebSEAL capability covering both internal and external users.
- There is a relatively small number of users initially, but this will grow dramatically.

We also know the following:

- All Internet access will go through a single site (San Diego).
- All Web servers we need to access are housed at a single site (San Diego).
- Web servers reside in an Internet DMZ network.
- Production systems reside in a special production network.
- All internal users share a common intranet across company site locations.
- Virtual host junctions will be required to accommodate the already existing Web site of ticker.stocks-4u.com.

From this, we can easily address an initial WebSEAL-based Access Manager architecture for Stocks-4u.com.

7.7 Access control architecture

As we know it today, the diagram in Figure 7-2 summarizes the existing security architecture deployed by Stocks-4u.com with multiple Web server host systems deployed in the Internet DMZ.

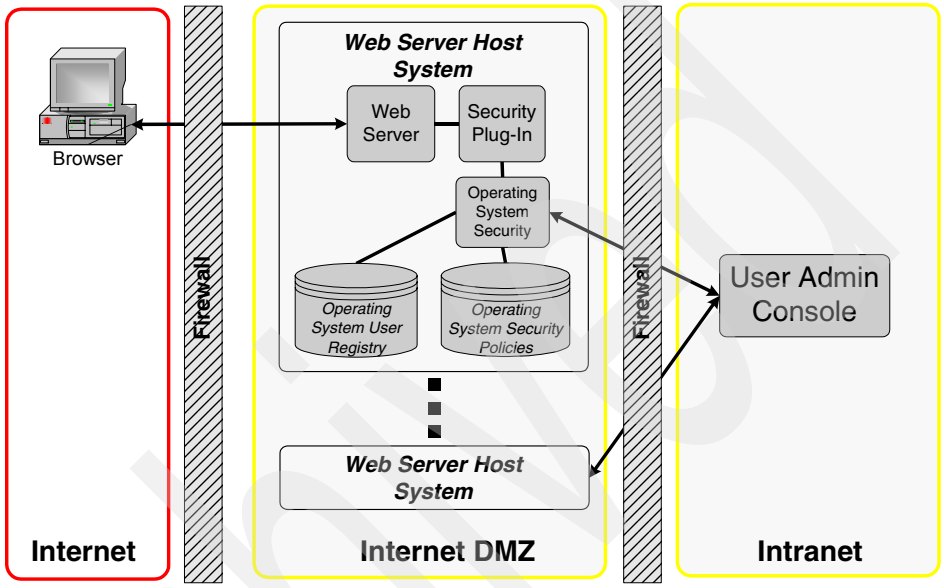


Figure 7-2 Current Stocks-4u.com architecture

These are the most pressing issues:

- The operating system security model is too centric.
- Key components are exposed within the DMZ.
- It is difficult to apply a uniform security model.
- Long-term maintenance staffing is required.
- It is difficult to keep up with evolving standards.
- Authentication is not flexible for requirements.

This is our starting point for developing an Access Manager architecture to meet current requirements, which are actually simple and straightforward, as we shall see.

7.7.1 Initial architecture approach

Recalling the discussions in Chapter 5, “Access Manager core components” on page 163, and Chapter 6, “Access Manager for e-business” on page 191, we know that we will place a WebSEAL server in the DMZ, which provides for

Internet user access. We also know that the user registry, Policy Server, and Web Portal Manager (WPM) should not reside in the DMZ. The user registry and Policy Server will be placed in the management zone while the WPM will reside in the production zone.

The company currently has its Web servers in the DMZ. With WebSEAL, there is no longer a need to do that, and these Web servers may be migrated to the production network. This is good because it enhances the security of the overall solution by moving the front-end application logic out of the DMZ.

Figure 7-3 displays our initial architectural diagram.

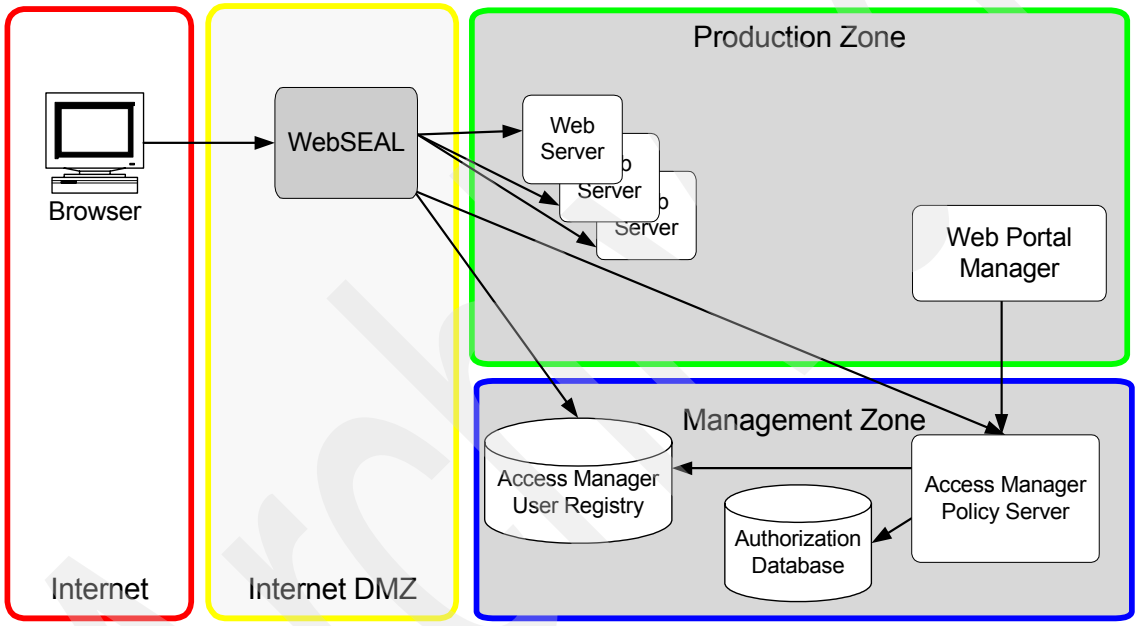


Figure 7-3 Initial WebSEAL architecture

This initial architecture provides us with the following benefits:

- The security model is independent of the operating system.
- We have a limited component exposure within the DMZ.
- It is architecturally consistent and we have a uniform security model.
- It is not dependent on internal resources to support core security component code.
- As standards evolve, the security infrastructure may be upgraded readily.

7.7.2 Internal user access

There are potentially many issues regarding internal user access, but for the moment we know that we only need to support employee access to internal applications from inside the company. In other words, Internet application access is currently only being provided for client applications and content.

We could route browser traffic to internal applications through the same WebSEAL that resides in the Internet DMZ. However, this is not a recommended approach, partly for security reasons, and partly for manageability and performance reasons. So in this case, we go with another WebSEAL server that is dedicated solely to internal access. This enables us to create a different set of junctions for the internal and external WebSEAL servers, which permits better segregation of content between the two access classes.

Tip: There may be scenarios in which it makes sense to have different user namespaces for employees and clients. This can be accomplished easily by creating a second Access Manager secure domain. However, in our scenario, such requirements do not exist. In this architecture, we will keep it simple and use a single Access Manager user registry covering both employee and client users in a common user ID namespace.

Where should this internal WebSEAL server reside? In our case, based on the Stocks-4u.com network structure, the logical place for this is in the production network. Figure 7-4 depicts the updated architecture diagram.

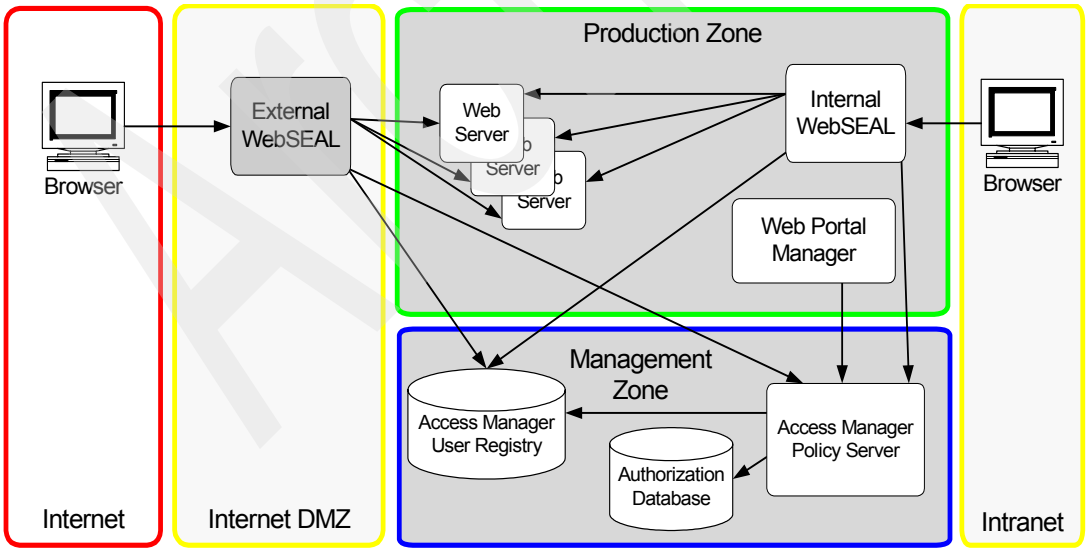


Figure 7-4 WebSEAL security architecture with internal WebSEAL

7.7.3 Connecting the pieces

Now that we have placed the key components in this scenario, we discuss how they interact with each other.

The Internet-facing WebSEAL will be listening on ports 80 and 443 (SSL). We will also modify the configuration of the Web servers slightly to have them listen on alternate ports (in our case, we use ports 81 and 1443). This enables us to close ports 80 and 443 on the firewall between the DMZ and production networks in the manner described previously in Chapter 6, “Access Manager for e-business” on page 191. We also disallow LDAP port (389/636) access from the Internet, because WebSEAL is the only entity that communicates from the DMZ to the user registry.

A virtual host junction will be created to allow ticker.stocks-4u.com to participate in the new Access Manager secure domain. By using virtual host junctions, we allow the ticker.stocks-4u.com name to remain and avoid having to immediately implement any changes to the application that would be required with a traditional junction. Although there will be two names registered in DNS that point to the WebSEAL server (stocks-4u.com and ticker.stocks-4u.com), the architecture does not need to be modified. One WebSEAL server can still meet these requirements.

There is also the question of whether the junctions between the Internet-facing WebSEAL and the Web servers require the use of SSL. It is not strictly necessary to do so in this case because the Web servers are in a controlled zone. If the Web servers were in the open corporate intranet, SSL should probably be used. The choice to use SSL may be made based on the specific risk associated with the content involved. The answer is similar with respect to communication with the user registry.

The internal WebSEAL in the production network, unlike the Internet-facing WebSEAL, will be co-located with the Web servers it is junctioned to. It will listen on ports 80 and 443, and the firewall between the intranet and production network will be configured to disallow access via these ports. If, for some reason, it is not possible to disable these ports (for example, there could be Web servers that are separate from the Access Manager infrastructure), the junctioned Web servers may be configured to accept connections only from the WebSEAL server. This would enable both WebSEAL and non-WebSEAL controlled resources to coexist in the same network while maintaining the integrity of the back-end Web servers.

Important: If you place a production Web server under WebSEAL access control, it is recommended that you do not allow access to it via non-WebSEAL channels without careful consideration. Prior experience has shown that this can lead to confusion, manageability issues, and most important, security breaches.

Generally, co-locating internal WebSEALs with Web servers is acceptable to many organizations; however, groups that may want to impose an internal DMZ in front of a production network may do so in the same manner as is done for the Internet-facing WebSEAL. This is a legitimate architecture and may make sense in some cases. However, in the current scenario, the requirements may be satisfied as we have described.

Now that we addressed the communication among the components, our new architecture is shown in Figure 7-5.

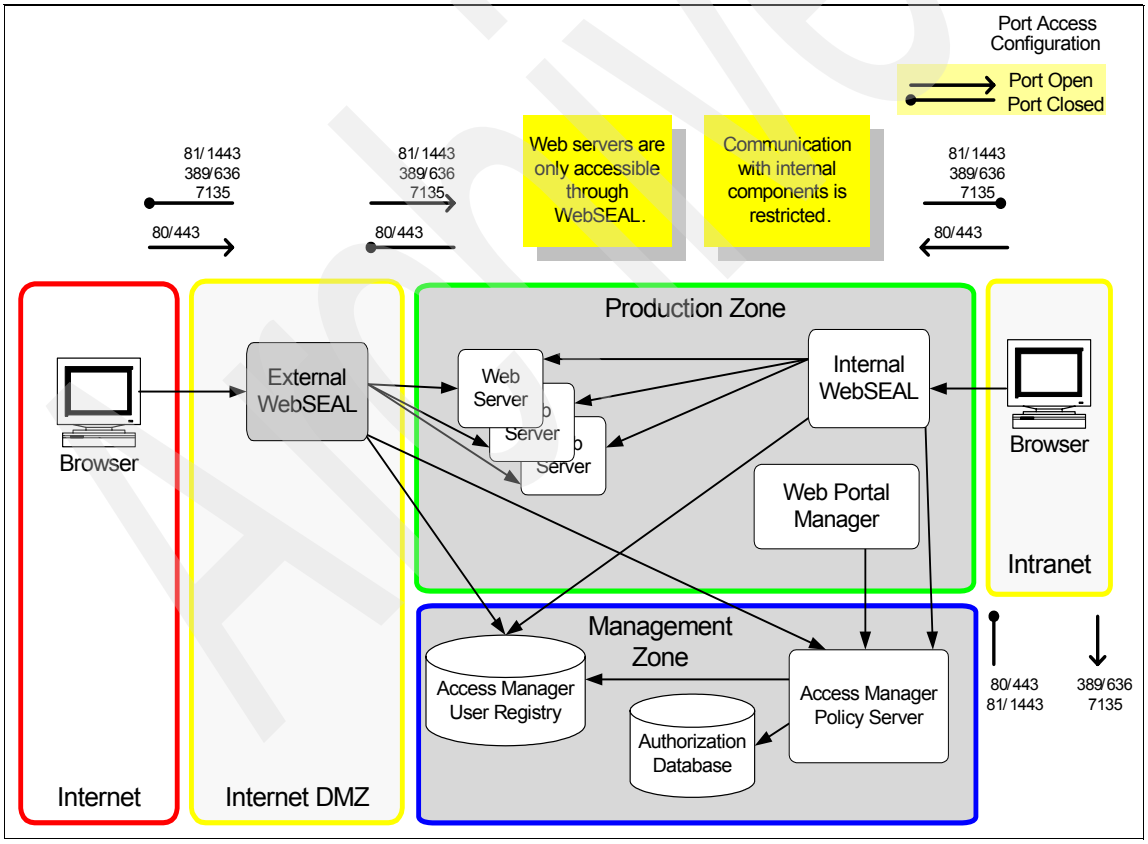


Figure 7-5 Detailed WebSEAL security architecture with internal WebSEAL

7.8 Building the physical architecture

With the locations of the pieces decided, now we conclude how many machines we need and what parts have to be configured on what systems.

7.8.1 Internet DMZ

Obviously, because the Internet-facing WebSEAL is in the DMZ by itself, it must be on a separate machine. This is typical for most WebSEAL scenarios. While technically this machine could support other applications or services along with WebSEAL, such configurations are not generally recommended, especially in an Internet-facing scenario.

A single WebSEAL host, appropriately configured, should be able to handle the expected client load over the next six months.

7.8.2 Production network

In the production network, things get only a little more complicated.

An obvious place to consolidate components would be to put the Access Manager Policy Server and the User Registry on the same machine, provided it has sufficient capacity. The policy server uses little overhead in a basic deployment such as this one, which has a relatively small number of components and users. The user registry is the major user of memory and processor capacity. We will place these components on a single machine.

Tip: However, it is important to point out that, as the company expands its operations, it may make sense to eventually split these functions onto separate machines. This should be easy to do when the time comes.

The WPM component can run on a Windows 2000/2003 platform as well as on AIX, Solaris, HP-UX, and Linux. One thing to keep in mind is that a midrange desktop system that meets minimum WebSphere memory requirements will generally work well to host WPM.

The internal WebSEAL is the remaining issue. Unlike the Internet-facing WebSEAL, we have more flexibility here. First, we know that the number of users is relatively small. However, they each perform several transactions per day. It may be possible to consolidate this WebSEAL onto the same host running the user registry and policy server. However, in this case, we opt to place the WebSEAL on a separate machine to avoid any potential performance effects due to component interactions.

Figure 7-6 shows the final physical architecture for the initial deployment.

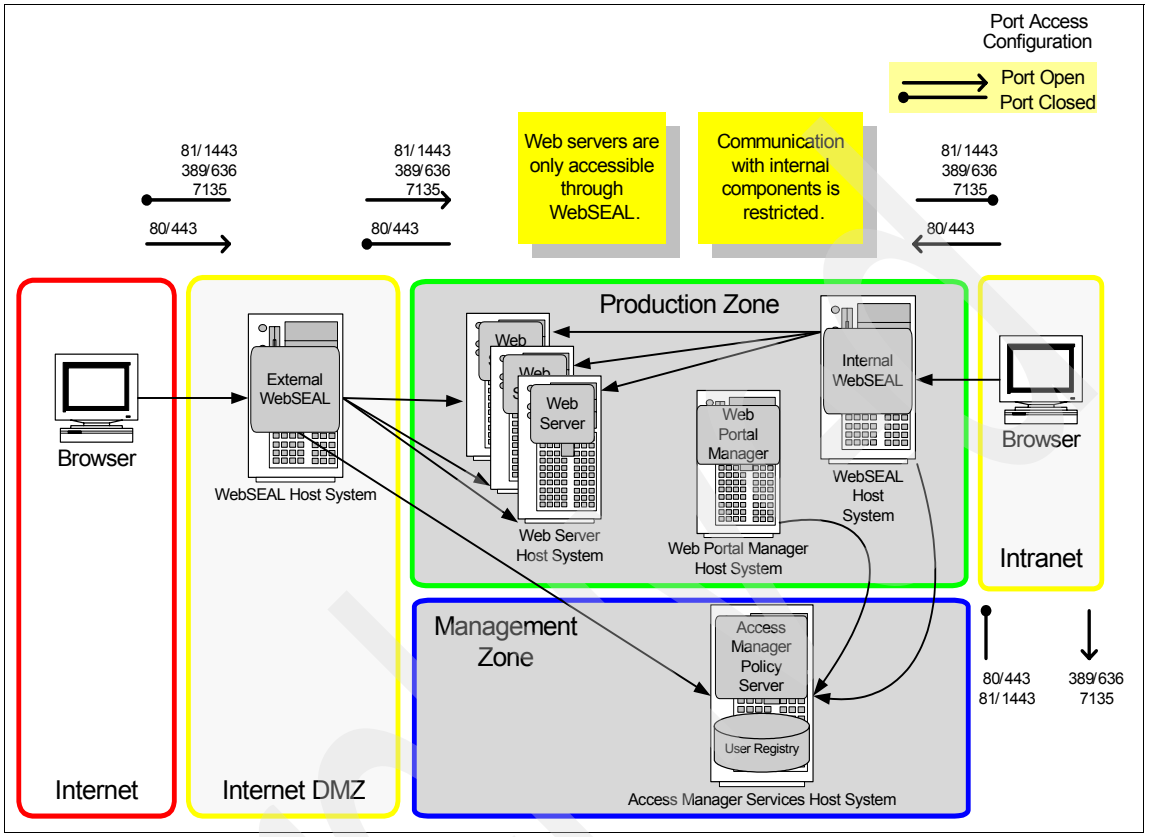


Figure 7-6 WebSEAL physical architecture

7.9 Architectural summary

In this chapter, we used the guidelines discussed previously in this book to illustrate the thought process involved in developing a typical WebSEAL solution architecture. You can understand that a Web security solution with Access Manager is often straightforward.

With this as a base, we can easily extend any Access Manager architecture to add additional capability and capacity, as we will see in later chapters.

8

Increasing availability and scalability

In this chapter we continue the discussion from the previous chapter with our customer Stocks-4U.com. Previously, the concern was access control and user and account integration, as well as systems and network integration. Now the focus has shifted slightly and the need to address additional requirements of a growing business have come to the forefront. This growth and the increased expectations pose new challenges to the architecture.

Availability is the major concern that a failing part of the infrastructure will cause the overall solution to languish. This eventually leads to unsatisfied customers and decreasing business success.

Scalability describes the ability to instantaneously change and adapt the IT infrastructure in order to handle an increased number of information and transaction requests without reducing the quality of the online experience for customers.

8.1 Further evolution

Stocks-4U.com has seen steady growth of their business. This growth, and the continued success of the business, has introduced new business requirements that mirror the evolving business. Based on these new requirements, we have to alter the security design objectives.

You, as the architect, now face the added design objectives of availability and scalability. Content, access control, and centralized audit and policy enforcement, as well as a single entry point into the site, are still very much a part of the scenario and must be included with the new requirements.

8.1.1 Business requirements

After the initial Web presence approach, the Web-based functions have functionally extended into content and applications and the security management becomes more viable. With the successful reception by the public, and an increasing client base, the availability of the Stocks-4U.com Web site is crucial. E-businesses have no set hours of operation and must be reachable and operational 24 hours a day, seven days a week (24x7).

At this stage, the CIO is looking for a way to guarantee the availability of the business application around the clock. Customers are entrusting their financial investments more and more to Stocks-4U.com, and they have to be rewarded with a reliable e-business application infrastructure that is always there for them.

After some serious downtime of the WebSEAL server (because of some operating system problems and issues with the back-end Web server availability, due to security vulnerabilities), the CIO demands that some protection measures in the availability and portability of the corresponding systems be taken.

A second concern of his is the constantly increasing number of customers visiting the Web site. The CIO asks for future flexibility and ways to dynamically add functional empowerment of the single systems to better cope with new e-business opportunities.

8.1.2 Security design objectives

The major design objectives of these business requirements target two areas of the e-business implementation:

- The access control infrastructure

  Embracing the internal and external WebSEALs, as well as the underlying security base, with the Access Manager Policy Server and the LDAP user registry

- The e-business application

  Consisting of the HTTP Web servers and the applications running on those servers

We have to consider two different approaches, as outlined by the CIO:

- Availability

  Enabling systems to be available on a 24x7 schedule by providing enough resources in additional, duplicated systems or other failover mechanisms.

- Scalability

  Enabling the e-business solution to scale to any number of future capacities by adding additional components of the same sort and providing smart load balancing mechanisms to perfectly utilize these new components. In a second viewpoint, this can also imply moving a current functional implementation to a new, more powerful operating platform.

8.2 Availability

The Internet has changed forever the idea of fixed hours of operation. Now there your customers expect to access your site at any time, day or night, increasing your visibility and profitability. The IT systems must be reliable and offer consistent content to the client in a timely fashion at any time. In our initial architecture, there are different points of failure in the infrastructure.

Each element in a configuration must be analyzed for failure points, including the hardware. Most hardware appliances, such as routers or switches, can be configured for failover or alternate paths, and cold standbys can be kept available, in case a hardware failure occurs.

The discussion in this section focuses on the availability of all components that are part of the Web application. We do not consider infrastructure elements, such as firewalls and routers.

8.2.1 Failure situations

Web servers and applications can and do fail. The reasons for failure vary: Program code, unproven technologies, disk failures, and even human error. In Figure 8-1 on page 262, the instance of only one WebSEAL user registry Access Manager Policy Server with its authorization master database, Web Portal Manager and each individual Web server are in themselves single points of failure.

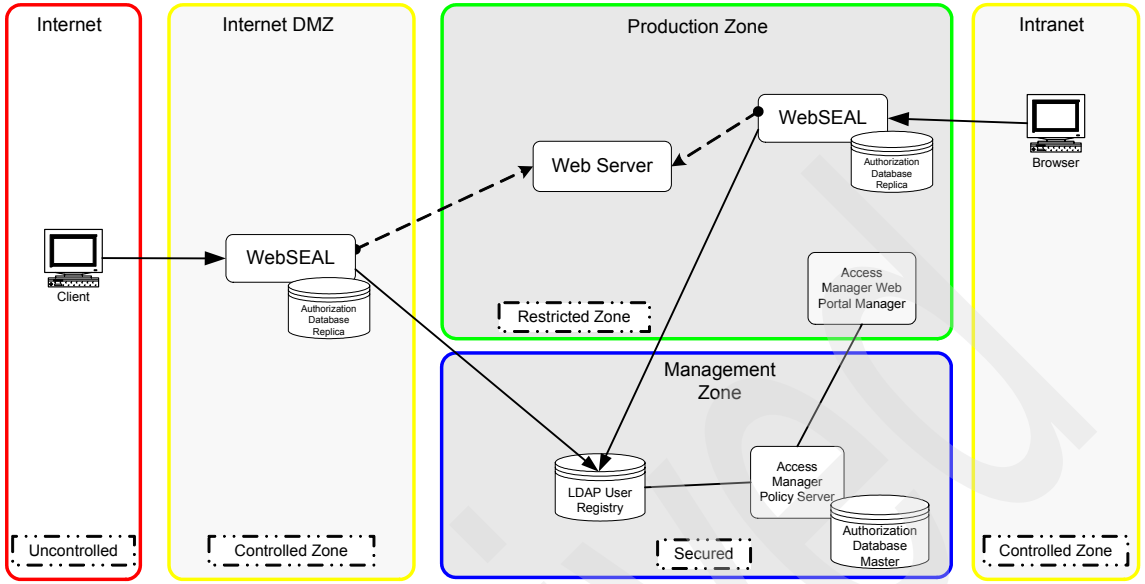


Figure 8-1 Initial Web architecture

What happens if the WebSEAL server fails? What happens if a Web server fails? What happens if the user registry server stops working? We now take a closer look at the individual components.

WebSEAL failure

If the WebSEAL portal to either the Internet or the intranet fails, and there is no operational replacement, the client attempting access will be denied access to the site. While the content and the application might be fully functional behind WebSEAL, the failure of the WebSEAL server leads the user to believe that the site is down.

Web server failure

If a Web server stops operating, the applications and services that reside on it are no longer available. While other applications are still working, the client that tries to access offerings on this particular machine perceives that the site or the application as down.

User registry failure

If the user registry is down, WebSEAL will no longer be able to authenticate incoming users in order to access Web content and applications that are protected and require user authentication. WebSEAL and the Web servers may

still be operational, but the client is unable to gain access and thus assumes that the site is down.

Access Manager Policy Server failure

Although failure of your Policy Server is not on your *wish list*, it does not affect the availability of your Web site. Web security servers can still perform all necessary authorization operations because they use the local cache mode, which means that the Authorization Service running on the WebSEAL machine uses a local authorization database replica. You only lose the ability to administer your Access Manager secure domain while your Policy Server is down.

The same is valid for the Web Portal Manager, which provides the administration graphical user interface Web application for the Access Manager administrators. The Web application will not be affected if WPM is not available. The only impact is that the administration of the Access Manager secure domain has to be postponed until the service is available again.

In addition to problems or failures of these components, sheer volumes can affect availability as well. With the growth of the Internet and your business, the ability to handle the traffic to your site has changed the scope and appearance of the architecture. Internet sites can become unstable or even fail under severe load conditions.

Tip: Besides adding multiple replicas for increasing availability and performance, you should also consider that your Web environment can scale on different operating system platforms with different availability characteristics. If you are stuck with only one supported platform, you might lose the ability to grow your business later.

The best example is the Web server itself. The IBM HTTP Server or the Apache Web server can scale from entry platforms such as Windows 2000/XP or Windows 2003 to other powerful platforms such as Solaris, HP-UX, AIX, Linux, or even OS/390® or z/OS. You should consider developing your Web applications supporting only open standards such as basic HTML, Java, Java Server Pages (JSP), or Enterprise Java Beans (EJB™); otherwise, you might get stuck with one particular platform.

8.2.2 Providing high availability

Adding replicas of crucial servers increases your site's availability. After depicting an overview of this configuration in Figure 8-2, we describe the different areas with their solutions.

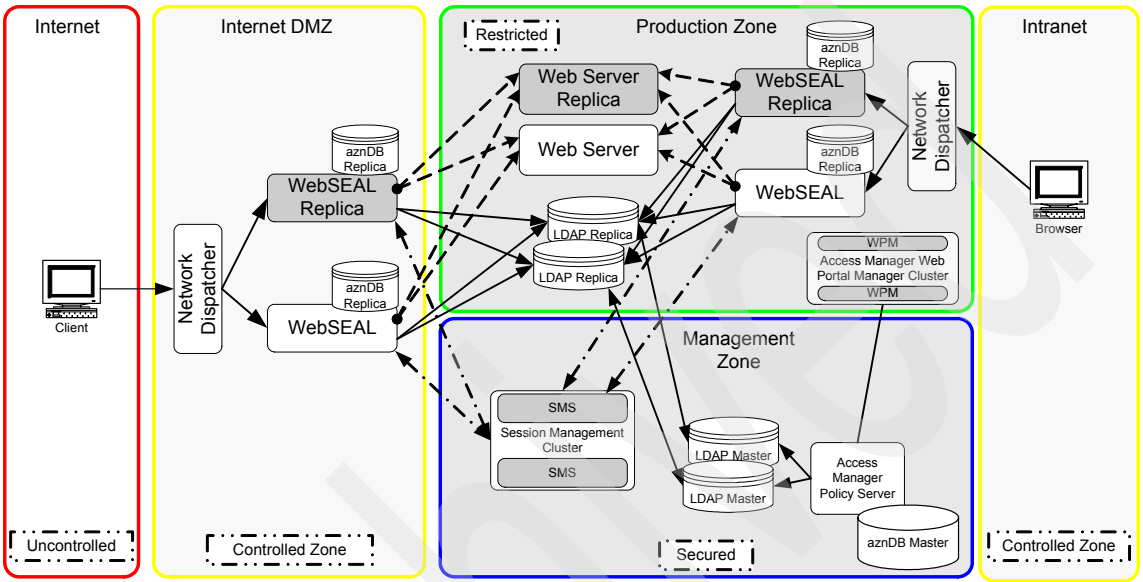


Figure 8-2 Server replication to increase availability

Session availability

It is recommended that the Tivoli Access Manager Session Management Server (SMS) be used in any situation where Web security server (WebSEAL or Web Server plugins) replicas are used. This prevents a user's originated session from being destroyed when the originating authentication service becomes unavailable. Although technologies, such as failover cookies, are available to provide seamless single sign-on to replica services, the replica session will not exactly resemble that of the original. With the use of the SMS, the replica services have access to the user's original session information to be used when this failover event occurs.

In conjunction to run time management of user sessions in a cluster or set of clusters, the SMS also collects user authentication statistics from across the cluster so that applications can maintain and display login history from within the Access Manager environment.

The Session Management Server overcomes obstacles in relation to session management in a clustered environment that include limitations for policy

enforcement, management, security, and the user experience. It also provides single sign-on between Web security servers in a failover situation.

Important: If the Session Management Server is used, failover cookies should not be used as the functionality is provided by the Session Management Server. There are rare cases where these functions can be used together; however, we recommend that you typically choose between the two.

The Session Management Server provides the following benefits in a clustered environment:

- Distributed session cache to manage sessions across clustered Web security server environments.
- Central point for maintaining login history information.
- Inconsistencies resolved between replicated Web security servers in regard to session inactivity and session lifetime time-outs.
- Single sign-on and secure failover among replicated Web security servers.
- Maximum number of concurrent sessions enforced across replicated Web security servers.
- Single sign-on capabilities among other Web security servers in the same DNS domain.
- Performance and high availability protection to the server environment on the event of a hardware or software failure.
- Administrators can view and modify sessions on Web security servers.

As the Session Management Server runs within WebSphere Application Server to provide high availability, it is typical for a customer to use WebSphere Clustering. More information about WebSphere clustering is available on the Web at the following location:

`http://www.ibm.com/software/webservers/appserv/was/network/`

Centralized account lockout information

Each Web security server can be configured to store a user's authentication failure count within the Access Manager user registry. By enabling this feature, all Web security servers can provide a consistent implementation of security policy since all failure count information is located in a central directory. This configuration is separate to that information stored within the Session Management Server, which is used for reporting purposes rather than policy enforcement.

Note: This functionality does not apply if using a custom external authentication C API program or External Authentication Interface (EAI). It can only be used if WebSEAL or the Web Plug-in is configured with the default user ID and password authentication module.

Also, in comparison to read operations, writing to an LDAP directory is an expensive operation. An evaluation of the environment's required and expected performance should be completed prior to activating this feature.

WebSEAL availability

Increasing the availability of your WebSEAL-controlled Web site starts with at least two front-end WebSEAL servers. Replicated front-end WebSEAL servers provide the site with load balancing during periods of heavy demand as well as failover capability. That is, if a server fails for some reason, remaining replica servers continue to provide access to the site. Successful load balancing and failover capability results in high availability for users of the site. The load balancing mechanism is handled by a mechanism such as the Network Dispatcher component of the IBM WebSphere Edge Server or Cisco Local Director.

In a redundant WebSEAL configuration environment, as depicted in Figure 8-3, there are several places where the configuration must be duplicated.

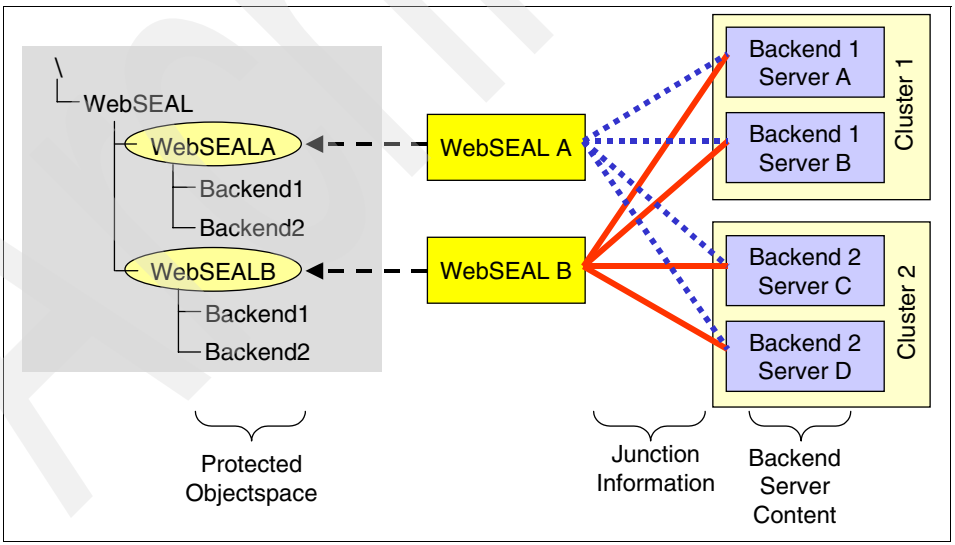


Figure 8-3 WebSEAL availability overview

- Back-end server content

  This must be the same on every server in the same cluster. Maintaining this is the responsibility of the individual system's administrator for the corresponding servers. More information can be found in "Web server availability" on page 268.

- Junction information

  Each duplicated WebSEAL server must have the same junction information. This is made easy in Access Manager because all that is required is copying the junction database from one WebSEAL to another. All junction information is kept in XML-formatted files.

- Protected object space

  Both WebSEALs must have the same ACLs attached to the same places in their object space. In a normal configuration, both WebSEALs have their own object space, so work must be duplicated. However, it is possible to make WebSEAL servers share a single object space.

WebSEAL clusters

In order to make two WebSEAL servers share the same object space, we change the part of the object space that one of the WebSEAL servers uses when making authorization decisions.

Normally, when WebSEALB checks permissions on objects, it uses its own unique objectspace. However, it is possible to have more than one WebSEAL check the same objectspace. This enables consistent application of security policy on multiple WebSEAL servers. However, in order for this type of setup to function properly, all WebSEALs that share an object space must have the same configuration and junctions.

Note: Be sure that a copy of the XML junction information is distributed to all clustered WebSEAL servers if new Web server junctions are being configured.

WebSEAL failover cookies

Failover cookies are used in Access Manager to enable a user to access a redundant WebSEAL server (in case of failure) without having to re-authenticate. Access Manager supports the use of failover cookies over HTTP or HTTPS. With the introduction of Access Manager 6.0, the Session Management Server component should be the preferred method of providing failover between replicated WebSEAL servers (refer to "Session availability" on page 264).

Note: The processing of failover cookies is processor-intensive and should only be used for failure recovery. In later versions of Access Manager, user's session id's are synchronized across Web security servers to minimize the affect of crypto expense when non-sticky load balancing algorithms are used.

More information about this configuration can be found in the section "Replicated front-end WebSEAL Servers" in the *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 6.0,* SC32-1687.

Web server availability

In order to increase the availability of your Web server space you have to duplicate your servers exactly. The Web administrator has to ensure that the content of the Web root directories on the duplicated servers is kept in sync. After you have created an initial WebSEAL junction for your first Web server, you can add your replicated Web servers to the same junction.

By default, Access Manager WebSEAL balances back-end server load by distributing requests across all available replicated servers when the replicated servers use the same junction point, as depicted in Figure 8-3 on page 266. Access Manager uses a "least-busy" algorithm for this task. This algorithm directs each new request to the server with the fewest connections already in progress.

For static Web content, this approach is very easy to implement. However, there are other considerations.

Maintaining a stateful junction

Most Web-enabled applications maintain a "state" for a sequence of HTTP requests from a client. This state is used, for example, to:

- Track a user's progress through the fields in a data entry form generated by a CGI program.
- Maintain a user's context when performing a series of database inquiries.
- Maintain a list of items in an online shopping cart application where a user randomly browses and selects items to purchase.

Servers that run Web-enabled applications can be replicated in order to improve availability through load sharing. When the WebSEAL server provides a junction to these replicated back-end servers, it must ensure that all requests contained within a client session are forwarded to the correct server and not distributed among the replicated back-end servers according to the load balancing rules. It maintains state through the use of a stateful cookie.

If the back-end application is capable of handling a failover event, WebSEAL should be configured to automatically associate the user's session with another server on the junction if one is available. This will allow a user to continue operating with no disruption of service. By default, this functionality is disabled.

Authorization Server availability

Although not initially depicted in the basic scenario in Figure 8-2 on page 264, assume for now that we extended our Web application using some fine-grained Authorization Application Programming Interface authorization calls. This authorization information is provided by Access Manager, and the application servers can be configured to request this information from a specific Access Manager Authorization Server if the applications run in remote cache mode configuration. This scenario is shown in Figure 8-4.

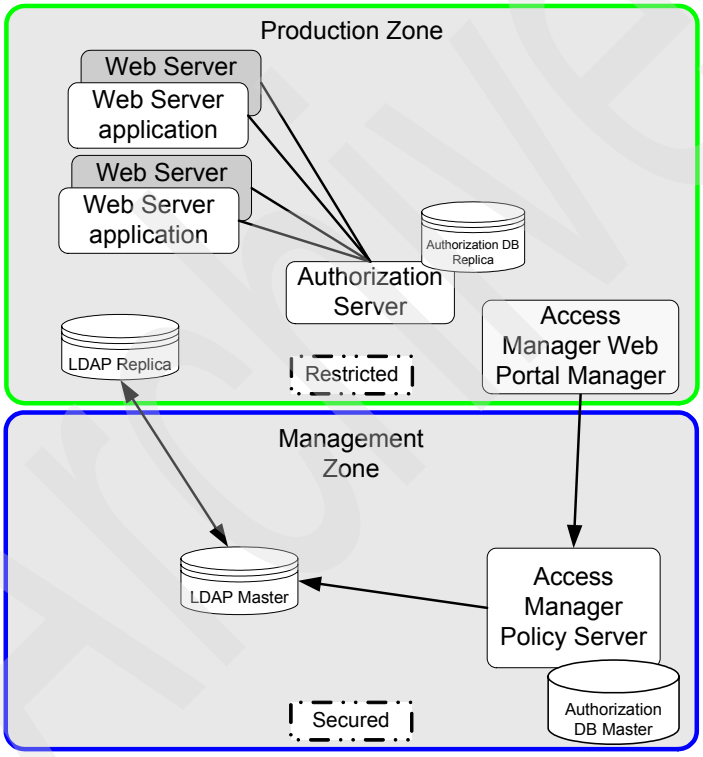


Figure 8-4 Authorization Server scenario for Stocks-4U.com

However, when this Authorization Server fails, the application cannot perform its fine-grained authorization calls and will therefore fail. In order to provide high availability of the application Authorization Services, the result would be the scenario configuration shown in Figure 8-5.

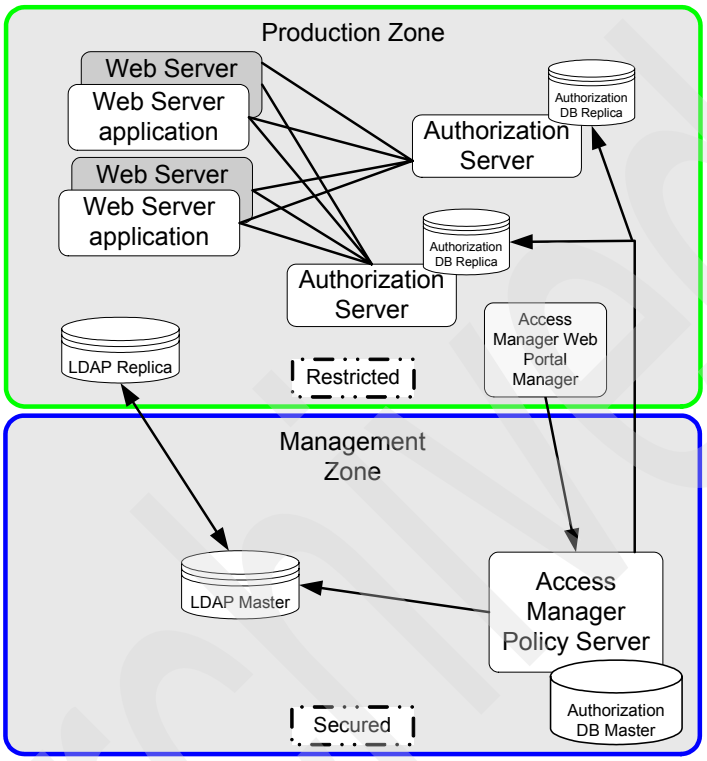


Figure 8-5 Authorization Server scenario with high availability

After implementing a second Authorization Server, you would only need to configure your aznAPI applications to be aware of the new replica.

Note: The second authorization server is only used for failover. Requests are not load balanced across the server instances.

Another way of implementing this particular scenario could be by configuring the applications to run in local cache mode, shown in Figure 8-6. By doing this, the aznAPI calls do not go out to a remote Authorization Server for access control checks, but instead uses the local authorization database replica.

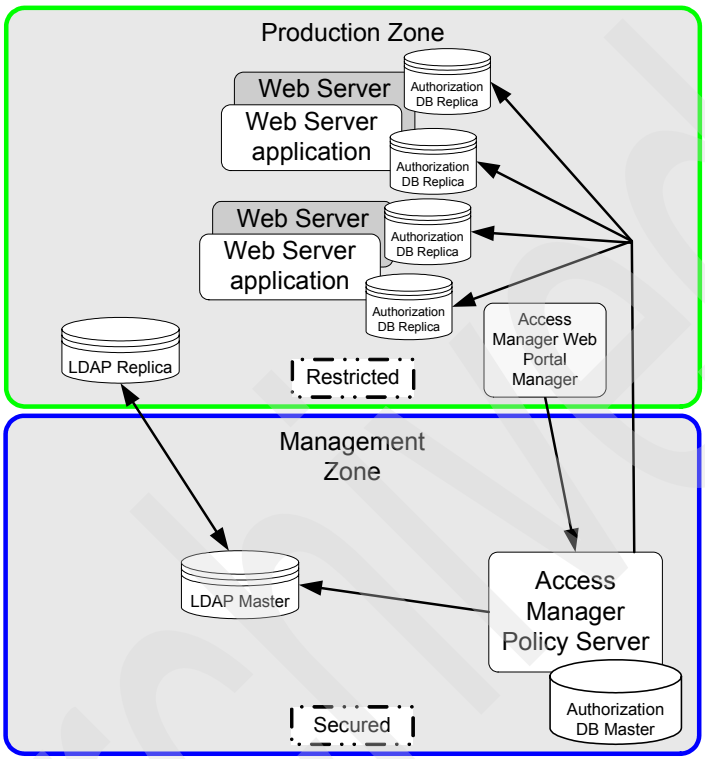


Figure 8-6 Authorization Server scenario on local cache mode

User registry availability

The IBM Tivoli Directory Server supports the concept of master and replica LDAP servers. This is discussed in more detail in 3.3, “IBM Tivoli Directory Server” on page 72.

A master server contains the master directory from which updates are propagated to replicas. All changes are made and occur on the master server, and the master is responsible for propagating these changes to the replicas.

A replica is an additional server that contains a database replica. The replicas must be exact copies of the master. The only updates that replicas allow are from replication from the master. The replica provides a backup to the master server. If

the master server crashes or is unreadable, the replica is still able to fulfill search requests and provide access to the data.

Access Manager utilization of multiple LDAP servers

The Access Manager connects to one of its LDAP master servers listed in configuration files when it starts up. At a minimum, the Access Manager server must be able to connect to an available LDAP replica server for any read operations.

Many operations, especially those from regular users, are read operations. These include such operations as user authentication and sign-on to back-end junctioned Web servers. After proper configuration, Access Manager LDAP server failover can be configured with priority given to replica servers over master (read-write) servers or to master servers over replica servers depending on the server being configured and the operations expected.

In order to configure Access Manager for the use of multiple LDAP directories, you have to define the master and replica LDAP servers to be used:

1. Master server configuration

   IBM Tivoli Directory Server and Sun Java System Directory support multiple read-write LDAP servers. Access Manager treats the Sun Java System Directory supplier server as the master server for configuration purposes. Access Manager can be made aware of multiple master LDAP servers through configuration.

   If you make a change to the LDAP database, such as adding a new user account through the WPM or changing a user's password through WebSEAL, Access Manager always uses the read-write (master) LDAP server.

2. Replica server configuration

   IBM Tivoli Directory Server supports the existence of one or more read-only replica LDAP servers. Sun Java System Directory Server supports the existence of one or more read-only replica LDAP servers referred to as consumers. Access Manager can be made aware of multiple LDAP replica servers through configuration.

   More about configuration can be found in the *IBM Tivoli Access Manager Version 6.0 Administration Guide,* SG32-1686.

Access Manager Policy Server availability

The only portion of Access Manager that cannot be replicated within the same secure domain is the Policy Server. You can, however, have a second server in stand-by to provide manual failover capabilities as a first aid response. If you want to assure 24x7 availability of your Access Manager Policy Server you could implement a high-availability cluster solution, such as HACMP for AIX. For

further details check the *HACMP Enhanced Scalability Handbook*, SG24-5328, and *Configuring Highly Available Clusters Using HACMP 4.5*, SG24-6845.

Before configuring a standby Policy Server, the files that it needs in order to operate must be made available. To avoid synchronization problems, it is best to locate these files on a shared filesystem.

In general, the most effective way to have a redundant Policy Server is to configure an *original* and *standby* Policy Server in an HACMP (or similar) environment. This handles routing IP traffic to the active instance and can handle (via scripting) the starting and stopping of the Policy Servers so that only one is active at any time.

Figure 8-7 shows a possible configuration that uses a network load-balancer to direct SSL traffic to the active Policy Server. If it is not possible for the load balancer to monitor the Policy Servers, then manual intervention (or custom scripting) will have to be used to monitor the Policy Servers and switch to the backup on failure.

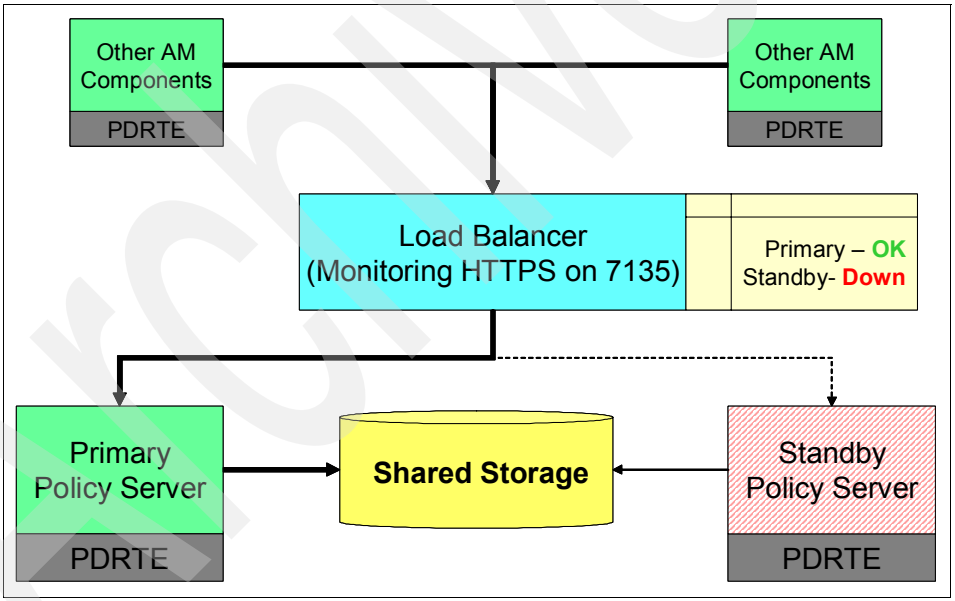


Figure 8-7 Standby policy server configuration using a load balancer

Note: The purpose of the Policy Server is to maintain the master authorization database that contains the protected object space with the access control information (ACLs, POPs, and Rules). The Policy Server replicates the authorization database to all other Access Manager Authorization Servers in the secure domain. Every application, configured in local cache mode, that uses this Authorization Service (such as WebSEAL and third-party utilization of the aznAPI) has its own local copy (replication) of the master authorization database and can therefore provide authentication and Authorization Services, even if the Policy Server is not available for a brief period of time.

Web Portal Manager availability

Again, the same is valid for the Web Portal Manager, which provides the administration GUI Web application for the Access Manager administrators. If the implementation requires a 24x7 availability of the Web administration interface, WebSphere clustering should be used to satisfy this requirement. Since Access Manager Web Portal Manager runs on the WebSphere Application Server, clustering of the application is supported by using WebSphere Application Server 6.0.2 Network Deployment. Configuring WebSphere Application Server for clustering is beyond the scope of this book. For more information about WebSphere clustering, refer to:

`http://www.ibm.com/software/webservers/appserv/was/network/`

Conclusion

Again, this point is clear: The Internet has changed the rules of how business is conducted. It has also changed the rules or concepts concerning customer loyalty. When users are experiencing slow response times or refused connections, they are having what is considered an unsatisfactory experience, which may cause them to never visit your site again and instead prefer one of your competitors. This line of thought leads us to the next discussion about scalability and performance.

8.3 Adding scalability

Scalability means that your systems have the capability to adapt readily to the intensity of use, volume, or demand. Designing scalability into your architecture also allows for failover of critical systems and continuous operation at the same time. A lot of the availability discussion can be applied to the scalability issue as well; the topics are all very similar. Here, we take a closer look at some specific viewpoints concerning scalability.

Access Manager automatically replicates the primary authorization policy database that contains the policy rules and credentials when a new application component, configured in local cache mode, or an Access Manager resource manager (such as WebSEAL or an Authorization Server) is configured. This capability provides the foundation of Access Manager's scalable architecture. After you have designed and installed your Access Manager secure domain and your Policy Server, you can easily extend and configure this IT security landscape.

8.3.1 WebSEAL scalability

To add additional capacity to a WebSEAL cluster, simply add another WebSEAL server behind an existing load balancer and configure it as a replica in the cluster. More information about WebSEAL clusters is in the previous section "WebSEAL clusters" on page 267.

The new WebSEAL will immediately receive browser requests that are routed from the load balancer product. This way, you can easily extend or change your WebSEAL infrastructure.

Tip: If you have installed WebSEAL multi-processor machines, they scale best if you put one WebSEAL per two CPUs, and lock them to use the specific CPUs only. Next, configure the WebSEAL instances into the load balancer.

8.3.2 Authorization Server scalability

To add additional capacity to the Access Manager Authorization Server infrastructure, simply install another Authorization Server and configure it as a replica.

The new Authorization Server will immediately be available to receive authorization requests from your applications. This way, you can easily extend your application infrastructure.

8.3.3 Infrastructure component scalability

In order to achieve overall scalability, we need to take a closer look at the other infrastructure components.

Web server scalability

When your current Web server-installed base is not capable of handling any more incoming requests, it is time to add a new server, maybe on a different,

more powerful hardware and operating system platform. To incorporate the new system into your existing Web server infrastructure:

1. Install a new HTTP server on a new machine and create an exact mirror of your published root directory structure from your existing Web server.
2. Add a WebSEAL junction to the same junction point as your existing Web server.
3. If you were previously using only one Web server at this particular junction, you have to consider defining a stateful junction at this time, if your Web application is relying on session states.
4. If you require SSL connections between WebSEAL and your Web server, you have to configure the junction appropriately.

Using WebSEAL as a mechanism for Web server load balancing and high availability makes it a simple task to scale your Web server environment to your individual demands. You could even replace a grown Web server cluster of multiple Intel machines with a new high-power server platform by reconfiguring your WebSEAL junction information, without losing one second worth of business or redefining any of your security access control information.

User registry scalability

In order to enhance the overall scalability of the implementation, LDAP master and replica servers can be added at will to improve the response time for user applications relying on LDAP access. In conjunction with using preference values, you can place LDAP replica servers close to the application functionalities—logically or location dependant.

Preference values for replica LDAP servers

Each replica LDAP server must have a preference value (1 through 10) that determines its priority for selection as:

- The primary read-only access server
- A backup read-only server during a failover

The higher the number, the higher the priority. If the primary read-only server fails for any reason, the server with the next highest preference value is used. If two or more servers have the same preference value, a least-busy load balancing algorithm determines which one is selected.

Remember that the master LDAP server can function as both a read-only and a read-write server. For read-only access, the master server has a hard-coded default preference setting of 5. This enables you to set replica servers at values higher or lower than the master to obtain the required performance. For example, with appropriate preference settings, you could prevent the master server from

handling everyday read operations. Access Manager can also load balance write operations in a multi-master LDAP environment.

You can set hierarchical preference values to allow access to a single LDAP server (with failover to the other servers), or set equal preferences for all servers and allow load balancing to dictate server selection. Further details about configuration can be found in the *IBM Tivoli Access Manager Administration Guide Version 6.0,* SC32-1686.

For further capacity and availability discussion, refer to the *IBM Tivoli Access Manager for e-business Problem Determination Guide Version 6.0*, SC32-1701, and the *IBM Tivoli Access Manager for e-business Performance Tuning Guide Version 6.0*, SC32-1704.

9

Authentication and single sign-on with Access Manager for e-business

This chapter describes the flexibility of user authentication mechanisms with Access Manager. It presents several mechanisms for the identification of users and shows how they can be used in various Web-based scenarios. It also introduces the basic concepts of achieving single sign-on solutions in Web-based environments.

This chapter does not look into any particular customer scenario, but rather presents the technological ground work for the scenario in Chapter 11, “Application integration” on page 347.

Different approaches are needed to provide different types of user access (for example, unrestricted access or restricted access with passwords, SecurID tokens, or PKI certificates) to a variety of back-end applications. This flexibility should be provided within one security solution, and the management of this security solution must support both centralized and distributed security administration groups, while maintenance of the Web applications can be done by other individual groups.

The goal of this security solution is to enable user authentication and to enforce target-based, coarse- or fine-grained authorization before forwarding a user's request along with his credentials to any of the Web application servers. This way, the Web application developers can stay free of maintaining any security infrastructures.

The represented Tivoli Web security solution is implemented as a reverse proxy Access Manager WebSEAL, which is located in the Internet demilitarized zone (DMZ). In order to serve as the single point of access control, it has to be used as the only access point for all incoming HTTP and HTTPS connections. Its major task will be to initially authenticate the user and to forward the user's request together with sufficient information about the user's identity to a Web server in a more secured network.

There are several issues we have to look out for:

- We have to make sure that WebSEAL does not allow any bypassing of the access control system. All internal and external access to Web-based resources should be channeled through WebSEAL.
- When using SSL connectivity to and from WebSEAL, you have to administer a private key for each WebSEAL and Web server participating in the SSL traffic flow. You should carefully control and document use of the private keys.
- You have to protect WebSEAL against unauthorized physical access. Because the reverse proxy has to terminate incoming SSL connections, all connection data will be unencrypted on WebSEAL. Although the data can be encrypted again when using an SSL connection to a back-end application server, physical access to WebSEAL or its memory might enable you to listen to communications even if the data is not being held in a cache.
- We recommend that you use a hardened operating system for WebSEAL. Do not use the machine for any other purposes. Restrict physical and logical access and use intrusion detection tools to monitor any type of unauthorized connection attempts.

We already focused on general WebSEAL architecture issues throughout Chapter 6, "Access Manager for e-business" on page 191, and Chapter 7, "A basic WebSEAL scenario" on page 245. In this chapter, we concentrate on the different authentication and single sign-on mechanisms that can be utilized with WebSEAL.

In addition to WebSEAL, the Web Server Plugins (referred jointly as the Web security servers) offer an approach for those customers who support the plugin model. The Web Server Plugins and WebSEAL product delivery teams endeavour to ensure that the product features are functionally equivalent. Obviously the architectural model is different, comparing the junctions of WebSEAL to that of the plugin model of the Web Server Plugins, but best effort

was made by development to make the functional aspects the same. In the chapters that follow, we reference Web security servers for referencing features that are shared between the two implementations.

9.1 Typical business requirements

In addition to the typical business requirements that were described in 6.2.1, “Typical business requirements” on page 193, which were driven by an overall Web security approach, we want to add the following concerns from the authentication aspect:

- The business application developers should only focus on business functions and not on security in order to eliminate hidden security management costs.

  Many applications use their own authentication and authorization mechanisms as well as security information repositories. There are also a lot of fields where basic operating system security is being used to achieve authentication. These approaches force applications to be maintained continually as changes to either security policy or operating system have to be implemented.

- Increase authentication flexibility without the need to change any application logic.

  Separate user registries for internal and external applications are used, as well as separate security administration for inside and outside applications.

  Another flexibility requirement is to allow different authentication methods for certain applications. A basic Web order system might be sufficiently protected with user ID and password authentication, while access to the same ordering system by business partners with high volume orders has to be controlled by providing a certificate-based or token-based authentication.

- Increase authentication strength within one session without the need to change any application logic.

  Sometimes it is necessary to process a step-up authentication when an already authenticated user tries to access data that is identified as critical. This would result in the user being prompted for an additional authentication after he already signed in.

9.2 Typical security design objectives

In addition to the typical security design requirements described in 6.2.2, “Typical design objectives (technical requirements)” on page 194, which were driven by an overall Web security approach, we want to add the following concerns from the authentication aspect.

Following are some of the technical requirements for authentication through the Web security servers address:

- Authentication

  Enforce authentication of users, where the type of authentication depends on the resources they want to access. Sometimes all users need to be authenticated, sometimes only users that want to access some protected URLs or applications need to identity themselves.

- User-based authorization

  Perform an initial user-based authorization check (such as, decide whether a user should be allowed to initially contact any of the Web applications). This step prevents certain users from accessing the system at all.

- Target-based authorization

  Perform a resource-based authorization by deciding whether a user should be allowed to contact a certain Web application.

- Single sign-on

  If user authentication and authorization was successful, forward the user’s request and user’s credentials to a certain Web application server for further processing.

- Use of a separate component for authentication

  It might be necessary to allow a separate and already existing authentication application and repository to perform the initial user authentication. These additional authentication methods should be usable without having to rewrite any of the applications.

9.3 Solution architecture with WebSEAL

The most secure way to achieve the design objectives is to use a reverse Web proxy with sufficient security functions in front of the existing Web application servers. Figure 9-1 shows a basic architecture for protecting Web applications.

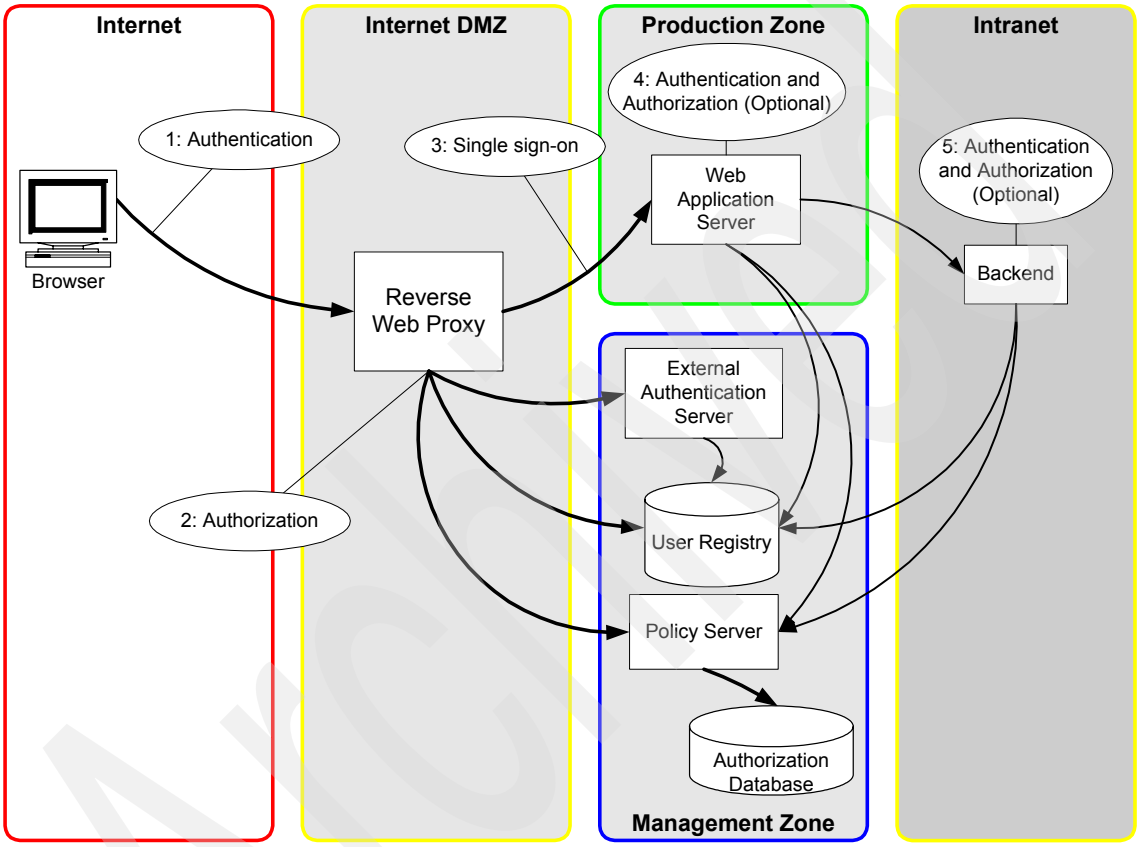


Figure 9-1 Reverse proxy flow for authentication, single sign-on, and authorization

The reverse proxy is used as a mediator between the user and the Web application servers. The functions of the reverse proxy have to provide the following details:

- Accept either HTTP or HTTPS connections.
- If needed, gather user credentials.
- If needed, perform user authentication (locally or by using an external authentication service such as an external authentication C API or external authentication interface (EAI)).

- Gather authorization information and make an authorization decision.
- Proxy the user's connection together with user credentials to the applicable Web application server.

Because this is a pure architectural discussion about functionality, the placement of additional components, such as load balancers and high-availability mechanisms, is described in Chapter 8, "Increasing availability and scalability" on page 259.

Note: The Web Server Plugins do not perform the SSL encryption between the client and the browser or the Web server and the applications (if remote) because this is the responsibility of the Web server itself.

9.3.1 Authentication and single sign-on mechanisms

This section presents the basic principles of authentication and single sign-on mechanisms that are used by the Web security server to enforce protected access when a user tries to connect to a certain Web application from its Web browser.

Authentication describes the process of exchanging credentials to identify the communication partners. Authentication can be directional or mutual. Single sign-on is the process of forwarding information about a user's identity in a secure way to another system. The Web security server can enforce certain types of user authentication and can use several single sign-on mechanisms to forward user requests together with user information to a Web application server.

Figure 9-2 on page 286 gives an overview of the various authentication and single sign-on mechanisms supported by the Web security server. It depicts the available authentication schema between a user and the Web security server, as well as the authentication between the Web security server and other back-end application servers. The different mechanisms are discussed in greater detail in 9.4, "Web security server authentication mechanisms" on page 291.

Look at Figure 9-1 on page 283 to follow the steps of the authentication process for a WebSEAL environment:

1. The user contacts the Web site by entering the HTTP address of a Web page or Web application. The first point of contact is the WebSEAL server. Because WebSEAL works as a reverse proxy, the user does not realize that

there is another system involved in the communication with the Web server that has been contacted.

If access to the requested information is restricted, WebSEAL requests authentication information and authenticates the user. After successful authentication, WebSEAL generates user credential information.

2. When authenticated, WebSEAL achieves an authorization decision based on the user credentials and the policy information that protects the information. WebSEAL decides whether the user is allowed to contact the system at all.
3. WebSEAL selects the junction for the user's requests and forwards the user credentials and user request to the Web application server.
4. Based on the forwarded user credentials, the Web application server can proceed with further, more fine-grained authorization decisions.

The Web security server solutions provide enough flexibility to support multiple authentication and single sign-on mechanisms to act as a reverse Web proxy between different user groups and different types of Web application servers in a secure way.

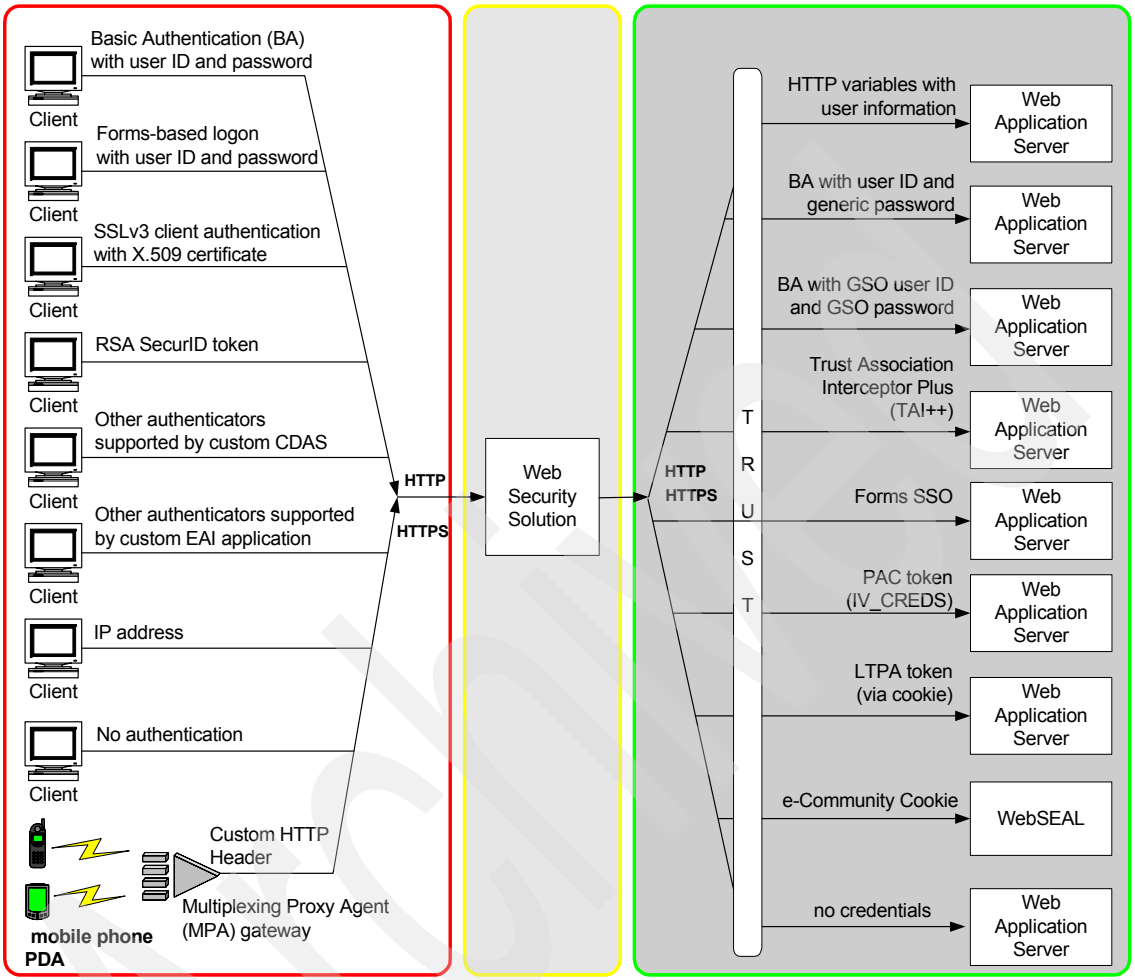


Figure 9-2 Access Manager authentication methods with Web Security Solutions

The left portion in Figure 9-2 lists authentication mechanisms available between a user and a Tivoli Web Security Solution. The right side lists single sign-on mechanisms between the Web security server and another Web application server. Typically the Web security server of choice here is WebSEAL because of its ability to provide downstream single sign-on capabilities.

Some of those mechanisms can be combined. For example, access to a certain URL can be restricted to require a certain IP source address and the correct user ID/password combination. It is also possible to combine some authentication mechanisms with single sign-on mechanisms.

A single Web security server may be configured for multiple different levels of authentication, of which unauthenticated is the first. Usually the next one is the user ID and password, but it can be any of the supported authentication mechanisms. Moving up to authenticated access happens when access control lists on the requested object do not allow access for unauthenticated users. The next level of authentication, which is usually a token (but can be any of the supported authentication mechanisms), is required when a protected object policy requiring it is set on an object.

9.3.2 Trust

An important factor for a centralized security portal solution is trust. If you configure all information requests to be routed through your central WebSEAL reverse proxy, you only want to authenticate the user once. This approach would imply that all back-end application servers trust all incoming user requests as being properly authenticated and authorized by a preliminary authority such as WebSEAL. This solution is very useful if WebSEAL can do all necessary authorization.

Figure 9-3 on page 288 shows a list of Web server products that can be protected with Access Manager's WebSEAL using some of the mechanisms that we listed. This list is not all inclusive. For the most up to date list of integration adapters visit the following download portion of the IBM Tivoli Access Manager for e-business Web site:

http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliAccessManagerfore-business.html

The topic of trust is less relevant in a Web Server Plugin discussion, where, typically, the plugin is only focussed on protecting access to a single application server (whether it be .NET or WebSphere).

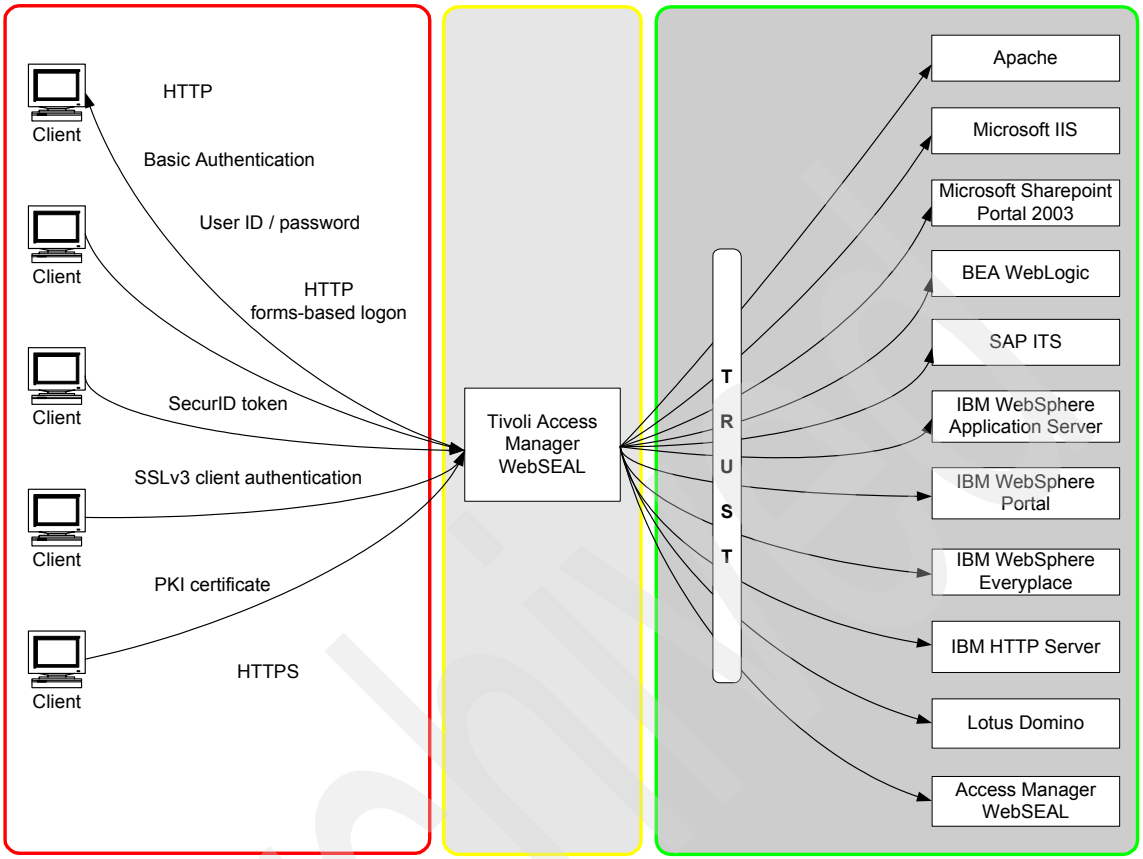


Figure 9-3 Overview of Web server products protected with WebSEAL

In order to fully implement a secure trust relationship, you would also have to configure each and every back-end application server to only accept incoming requests from WebSEAL on the specified port. No other direct connections, internal or external, are to be allowed to any of the servers. In cases where this is not yet practical or possible to achieve, you would have to specify the junctions to forward the user credentials in a way for the back-end servers to re-authenticate the user principal. This discussion was also addressed in Chapter 6, “Access Manager for e-business” on page 191.

9.3.3 Generic authentication mechanism with Web security server

Before going into the specific authentication model details, we use Figure 9-4 to look at a generic picture of the Web security server authentication model.

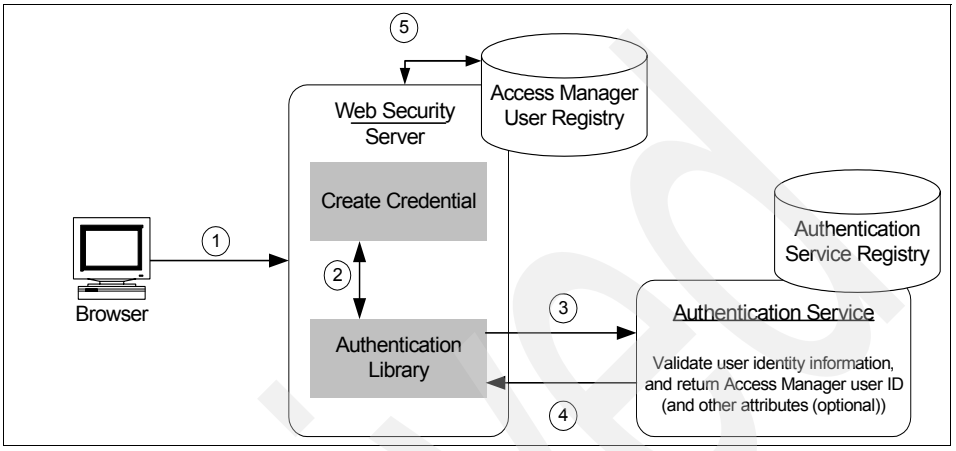


Figure 9-4 Generic WebSEAL authentication model

The following steps explain Figure 9-4.

1. The user presents his identity information to the Web security server.
2. The Web security server invokes the configured authentication library (password, token, certificate, or custom).
3. The authentication library passes the user identity information to the Authentication Service to perform user validation.
4. After validating the user, the Authentication Service maps the information according to its configuration and returns an Access Manager user ID. The Authentication Service may return the same individual user information that it received on input or it may use a mapped-to ID if each input user is also the output user that is referred to as one-to-one mapping. If many input users are mapped to the same output user, that is referred to as many-to-one. In both cases, the returned user must be defined to the Access Manager's user registry.
5. The Web security server now uses the Access Manager user registry to create the Access Manager credential that is cached for the duration of the session and used for any authorization decisions.

9.3.4 Generic Web security server single sign-on mechanism

As discussed in Chapter 7, “A basic WebSEAL scenario” on page 245, WebSEAL provides powerful capabilities for managing access to multiple Web application servers through a common access point. Figure 9-3 on page 288 shows some of the various products WebSEAL can integrate with. Some of these downstream single sign-on attributes can also be configured to be provided by the Web Server Plugins.

If you want to delegate further authentication and authorization tasks to the back-end application, you have to provide information about the user and the session. In order to pass on that kind of information, you have to define your junctions accordingly. You can actually provide the following information for your junctioned servers.

Supplying client identity in HTTP headers

You can insert Access Manager-specific client identity and group membership information into the HTTP headers of requests destined for junctioned third-party servers. The Access Manager HTTP header information enables applications on junctioned third-party servers to perform user-specific actions based on the client’s Access Manager identity.

Supplying client IP addresses in HTTP headers

You can insert the client IP address information into the HTTP headers of requests destined for junctioned application servers. The Access Manager HTTP header information enables applications on junctioned third-party servers to perform actions based on this IP address information.

Passing session cookies to junctioned portal servers

A Web portal is a server that offers a broad array of personalized resources and services. You can send the Access Manager session cookie (originally established between the client and the Web security server) to a back-end portal server. This option currently exists to directly support the integration of WebSEAL with different vendors’ portal solutions. Note that the passing of session cookies is for downstream SSO from the portal to applications and for performing session termination using applications built with the Management API.

Global Single sign-on solution

Access Manager supports a flexible single sign-on solution that features the ability to provide alternative user names and passwords to the back-end Web application server.

Dynamic business entitlements

Access Manager offers a dynamic business entitlement functionality for passing information to back-end Web applications. This is implemented with two steps:

1. It is possible to insert any field from an Access Manager user's LDAP record into the user's credential at logon time. These values can be extracted by an application using the Authorization Application Programming Interface accessing the delegated client identity information.
2. Being able to insert arbitrary values from LDAP into the credential (without writing new authentication code) is a useful addition to Access Manager; however, the next step goes one step further, enabling back-end Web applications to access the information without the need to use aznAPI.

   The Web security servers can extract the values from the credential and pass them to the back-end Web server as fields in the HTTP request header. This enables most Web applications to access them without using any special code.

9.4 Web security server authentication mechanisms

This section shows the authentication mechanisms that are supported by the Web security servers to protect access to a Web environment. Some mechanisms in this section can be combined with some of the single sign-on mechanisms in the next chapter to make the connection between a user and a Web application.

The Web security servers use the concept of authentication modules to use different authentication methods. There are three types of modules:

- Built-in modules that ship with Access Manager Web security servers and are fully supported
- Support for custom external authentication solutions using the *external authentication interface* (EAI)
- Support for custom modules written using the external authentication C API

The following built-in modules exist in Access Manager:

passwd-ldap	Password authentication via LDAP (Forms/BasicAuth)
passwd-uraf	Password authentication using the Tivoli Access Manager User Registry Adapter Framework (URAF) for Active Directory or Domino (Forms/Basic Auth)
token-cdas	Token authentication (SecureID)
cert-ldap	SSL client certificate authentication

http-request	HTTP header or IP address authentication
kerberosv5	Implements Simple and Protected Negotiation (SPNEGO) authentication with WebSEAL (Windows Desktop Single Sign-On)

9.4.1 Basic authentication with user ID and password

Basic authentication (BA) is part of the HTTP standard and defines a standardized way in which user ID and password information is passed to a Web server. When the Web security server sends a BA challenge to the browser, the browser pops up a dialog panel requesting user name and password from the user. When this information is entered, the browser sends its original request again, but this time with the user name and password included in the BA header of the HTTP request. The Web security server extracts this information from the header and uses it to verify the user's identity. In this case, a specific library shipped with Access Manager implements a built-in authentication service and performs a check against the Access Manager user registry. If successful, a credential is created and cached.

After a user has authenticated an ID and password through the browser, the browser caches this information in memory and sends it with each subsequent request to the same server. Even by configuring a session log-out parameter, which is possible for HTTPS sessions, the user automatically logs on to Web security server with each new request he sends. The only way to clear this cache (and log the current user out) is to close all browser panels.

9.4.2 Forms-based login with user ID and password

The alternative to using basic authentication is to use forms-based login. Rather than send a basic authentication challenge in response to a client request, Web security server responds with a sign-in form in HTML format. The client browser displays this and the user fills in a user ID and password. When the user clicks the send or logon button, the form is returned to the Web security server using an HTTP POST request. The Web security server extracts the information and uses it to verify the user's identity through the Access Manager authentication service, where it performs a check against the Access Manager user registry.

As the user ID and password information is not cached on the browser, it becomes possible to perform a programmatic logout for the user. On a client request, the Web security server presents a customizable logout form to a user. After the user confirms the logout, the session is considered closed and the credential is deleted from the Web security server cache.

Another benefit to using the forms-based login process is that you can enforce a time-based logout for authenticated sessions. The time values can be customized in the Web security server configuration files.

9.4.3 Authentication with X.509 client certificates

In response to a certificate request from Web security server, as part of the SSL Version 3 tunnel negotiation, the browser prompts the user to select a certificate from the local certificate store or smartcard. The user is asked for a password to access the private key. When the user selects a certificate, it is passed to the Web security server, which uses the certificate authentication library to check the signature of the client certificate. It also checks the validity period to ensure that the certificate has not expired. Assuming that the certificate is valid, the identity in the certificate is mapped (one-to-one) to an Access Manager identity. After the Access Manager identity is passed back to Web security server, the Web security server pulls the user information from the Access Manager user registry and builds the credential.

If you configure Access Manager to use X.509 client certificates for authentication, but the user does not have a certificate available, Web security server can fall back to basic authentication, if required.

9.4.4 Authentication with RSA SecurID token

Access Manager includes an external authentication C API that supports authentication of clients using user name and token pass code information from an RSA SecurID token authenticator (TAR), a physical device that stores and dynamically generates a piece of authentication data (a token).

The TAR is used in tandem with an authentication server (the RSA ACE/Server), which actually performs the authentication. During authentication to the Web security server, the client enters a user name and pass code. The pass code consists of the following:

- The unique PIN number associated with the client's SecurID TAR
- The current number sequence generated by the SecurID TAR

The Ace/Server uses its own registry database to determine the PIN that the user should be using, checks it, and strips it off of the pass code. It then checks the remaining number sequence against its own internally generated number sequence. A matching number sequence completes the authentication.

At this point, the role of the external authentication C API token is complete. The external authentication C API does not perform identity mapping, but simply returns to the Web security server an Access Manager identity containing the

user name of the client. This user name must match a user ID stored in the Access Manager user registry.

9.4.5 Windows desktop single sign-on

Before describing Windows desktop single sign-on, there are some important security considerations to point out:

- In order for Microsoft Internet Explorer® (IE) to be able to use integrated Windows authentication, it must recognize the Web security server server as an *intranet* or *Trusted* site.
- The Web security server must be able to access Active Directory as its Kerberos Key Distribution Center (KDC). This may expose Active Directory to new networks.

Therefore, it is important to only use SPNEGO authentication over a secure network or over a secure transport.

The Web security servers support the SPNEGO (Simple and Protected GSS-API Negotiation) protocol and Kerberos authentication for use with Windows clients to achieve Windows desktop single sign-on. The SPNEGO protocol allows for a negotiation between the client (browser) and the server regarding the authentication mechanism to use. The client identity presented by the browser can be verified by the Web security server using Kerberos authentication mechanisms.

Note: Use of SPNEGO requires that a time synchronization service be deployed across the Active Directory server, the Web security server, and any clients that will authenticate using SPNEGO.

Figure 9-5 on page 295 illustrates how the Web security server responds to requests when configured for SPNEGO authentication.

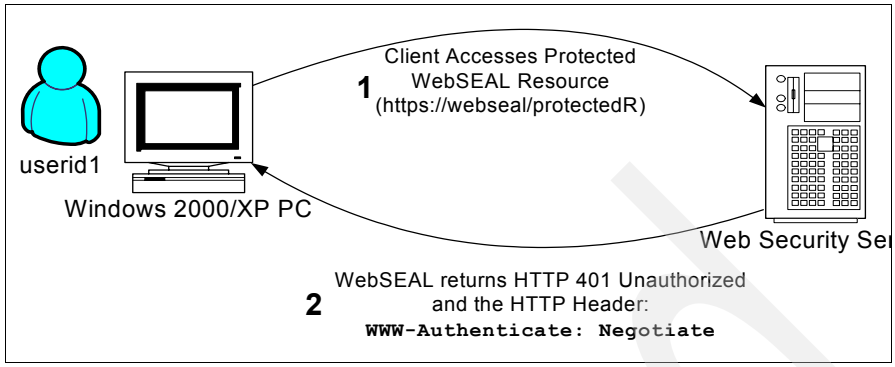


Figure 9-5 Initial negotiation with SPNEGO configured Web security server

The response provided above indicates that the Web security server includes the `WWW-authenticate: Negotiate` header in its response. In addition, the Web security server also sends the login form back to the requestor's browser. This allows for a user that is not enabled for SPNEGO authentication to still authenticate and participate in the Access Manager secure domain. If the user's browser is configured for integrated login, then the SPNEGO authentication process can begin. Figure 9-6 shows the SPNEGO authentication process.

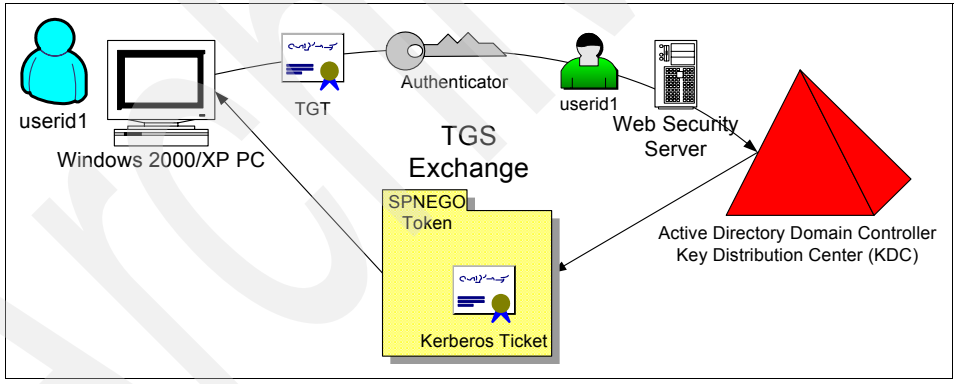


Figure 9-6 SPNEGO authentication process

Notice that in order to gain access to the Web security server, the user must first obtain a ticket from the Key Distribution Center (KDC), which is also Active Directory. This ticket is then sent in a SPNEGO message to the Web security server as show in Figure 9-7 on page 296.

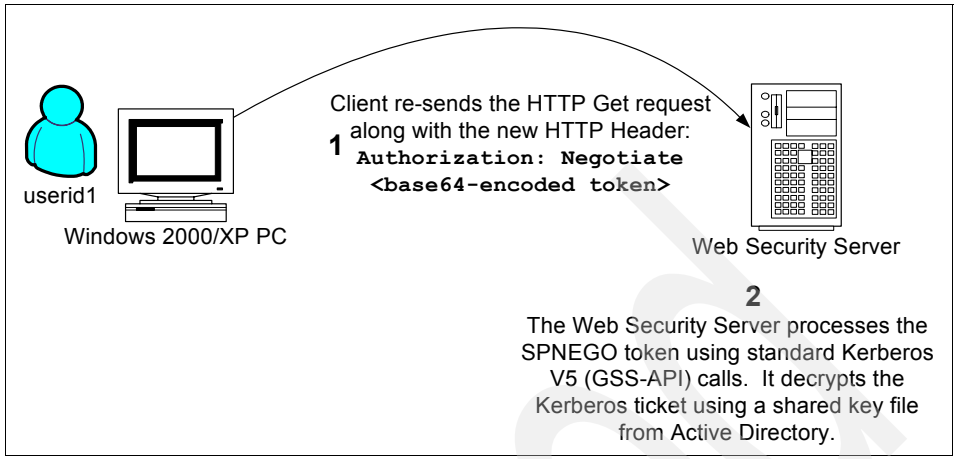


Figure 9-7 The Web security server processing of SPNEGO token

The last portion of the Authentication process is to build an Access Manager credential as show in Figure 9-8.

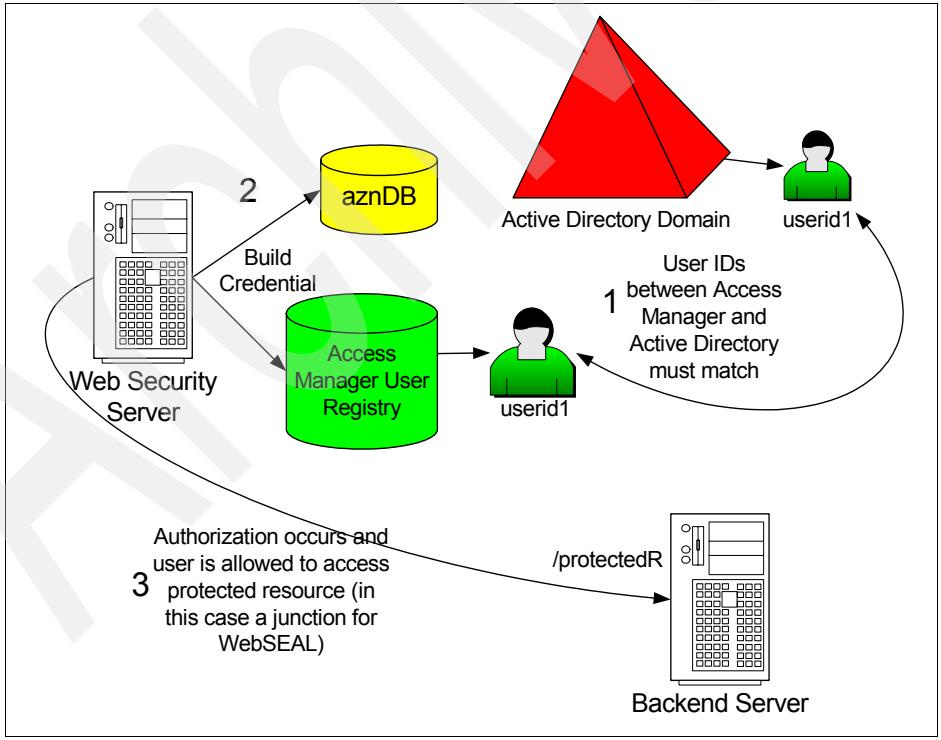


Figure 9-8 Building a credential after SPNEGO authentication

Mapping an ID from Active Directory to Access Manager is an important part of SPNEGO. Normally, the Web security server truncates the domain name portion of an Active Directory ID off in order to get the user ID. This can cause conflicts, however, if two different users have the same ID in different Active Directory domains. In this case, the Web security server would need to be configured to keep the domain section of the user ID attached in order to be able to resolve the conflict. Any more complex mapping scenarios of Active Directory IDs to Access Manager IDs requires a custom module to be written to modify the user name returned from SPNEGO authentication.

Also, if the desire is to have a user the IDs between Active Directory and Access Manager remain synchronized, an directory synchronization product such as IBM Tivoli Directory Integrator should be used.

Support for Kerberos authentication in the Web security server was implemented specifically to support a Windows desktop single sign-on solution. This solution requires that the Web security server server have accounts in an Active Directory domain, and that they be able to access a Kerberos Key Distribution Center. In addition, the Internet Explorer (IE) client must be configured to use the SPNEGO protocol and Kerberos authentication when contacting the Web security server.

Kerberos authentication, which uses Active Directory services, is supported by the Web security server running on Windows, AIX, Solaris, Linux on x86, and zSeries Linux.

Note: Compatibility between SPNEGO authentication and the Web security server e-community single sign-on is limited. The Web security server can be an e-community master authentication server (MAS) and support SPNEGO. However, the Web security server server cannot be an e-community subordinate and also support SPNEGO.

If you have to use the older NTLM (NT LAN Manager) authentication, which involves passing a token based on the user's local password, your only option is to use the Web server plug-in for IIS.

9.4.6 External Authentication Interface

Tivoli Access Manager Web security servers both support the externalization of authentication through an HTTP interface. This technology is known as the *External Authentication Interface* (EAI). The external authentication interface is an alternative way to customize authentication when the authentication information is passed in HTTP messages. It allows a backend application server to perform the authentication of a user (with the HTTP messages passing through the Web security server) and then, upon successful authentication,

return an identity to the Web security server using some predefined HTTP headers.

By allowing an application server to perform authentication, virtually any desired authentication strategy can be implemented. Since the interface is HTTP, the backend application can be written in any language that supports communication via the HTTP protocol. This is an advantage over the external authentication C API that must be written exclusively in C. The external authentication C API is still, however, the only method for performing non-HTTP authentication such as client certificate authentication.

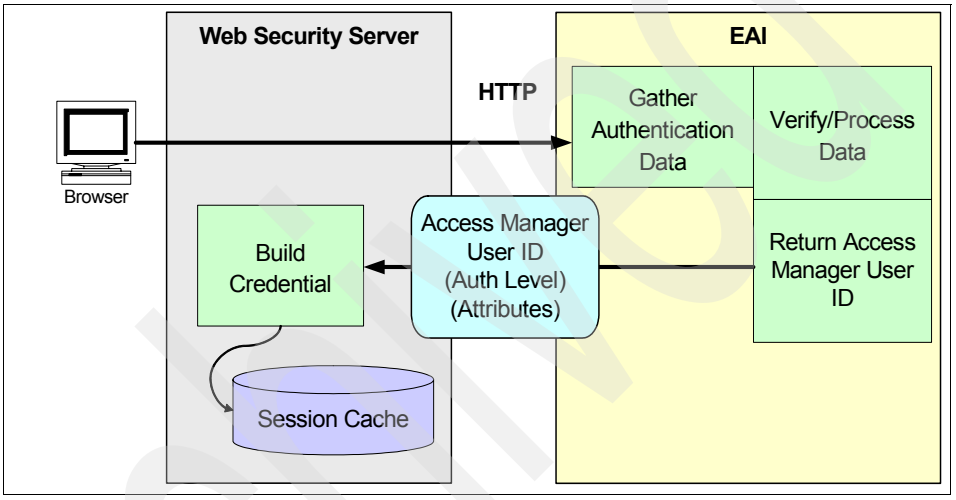


Figure 9-9 External Authentication Interface

The external authentication interface could return a Privilege Attribute Certificate (PAC) which could then be used to build the credential. It can also return identity components that the Web security server can use to create a credential. These include all components that can be returned from an authentication module, such as username, authentication level, and extended attributes.

There are some implications to externalizing authentication outside of the standard Web security server password module. The password module in the Web security server provides for maximum failed login attempts, password lifetime checks, and a password change function. All of these functions need to be duplicated if using an external authorization interface.

A benefit that comes with implementing an external authorization interface is that the restrictions on user registries for authentication is no longer applicable. In theory you could authenticate against any user registry you want (directory, database, and so on) provided that the interface you are writing supports it. In

addition, you could also authenticate against multiple user registries. Regardless of where the external authentication interface authenticates a user or how it authenticates them, the EAI must return a valid Access Manager user. This means that user and groups must exist in the Access Manager user registry that can represent users and groups in the foreign registry. There are three ways to approach this problem:

- Synchronize user registries

  User and group objects could be synchronized from the foreign user registry into the Access Manager user registry. This allows for user level authorization to still be performed within the Web security server. When the user ID is passed from the EAI to the Web security server, group information is pulled into the credential from the Access Manager user registry not the foreign user registry. Since authentication is not being performed against the Access Manager user registry there is no need to synchronize user passwords or password policy information. Since the synchronization would need to be constant as users and group could be modified on both the Access Manager user registry and the foreign user registry, IBM Tivoli Directory Integrator would be a good solution for user registry synchronization in this situation.

- Fixed user ID returned

  User and group information is not synchronized between the Access Manager user registry and the foreign user registry. The EAI returns a fixed user ID to the Web security server. While easier to implement, this solution has serious drawbacks in terms of enforcing security policy. Since all users being authenticated by the EAI are returning the same user ID to the Web security server, there is no way to use user dependent ACLs for security. This model simply allows for authenticated or unauthenticated access to resources. Authorization rules could be used to enforce policy, however, if the EAI included the actual user ID in an extended attribute in the credential. Using only authorization rules for security results in higher administrative overhead due to the effort needed to define the rules. It also results in lower system performance as evaluating rules is more expensive than evaluating ACLs.

- Dynamic group assignment

  This option only works if the EAI passes back a credential to the Web security server (this is also known as a Privilege Attribute Certificate or PAC). The EAI would insert group membership information from the foreign user registry into the user's credential. The groups could then be synchronized from the foreign user registry into the Access Manager user registry. Another way to perform this type of mapping is to have the EAI map the users into a specified set of static groups in the Access Manager user registry. Using this technique, authentication is performed against a foreign user registry and the group memberships in the foreign user registry can be reflected in the Access

Manager credential. ACL authorization can now be performed at the group level. It is important to be aware that user level authorization is still not possible since the EAI is still returning a fixed user ID to the Web security server.

EAI using IBM Tivoli Directory Integrator

Directory Integrator can be used as an external authentication interface into Access Manager, the Web security server. Directory Integrator has an HTTP server connector that allows an assembly line to listen for HTTP requests and then runs an assembly line with the HTTP request as input. The assembly line output sends an HTTP response to the caller. Figure 9-10 shows how Directory Integrator can be used as an EAI to perform directory chaining—users defined in multiple registries can all be authenticated to the Web security server.

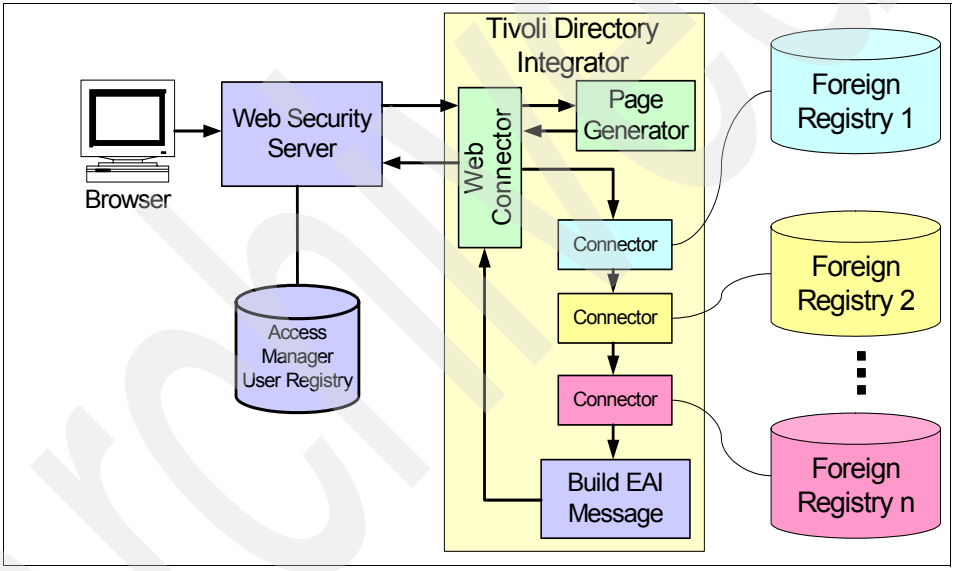


Figure 9-10 Directory chaining EAI using IBM Tivoli Directory Integrator

Another problem that can be solved using a Directory Integrator-based EAI solution is user migration of passwords. Figure 9-11 provides an example of a user authenticating to the original, foreign user registry using a Directory Integrator-based EAI application. The assembly line can then create a user ID for the user in the Access Manager user registry, including their password, before returning that new user ID to the Web security server. The Web security server can then build the user credential based on the newly created user. This whole process of user migration would be completely automatic and transparent to the end user.

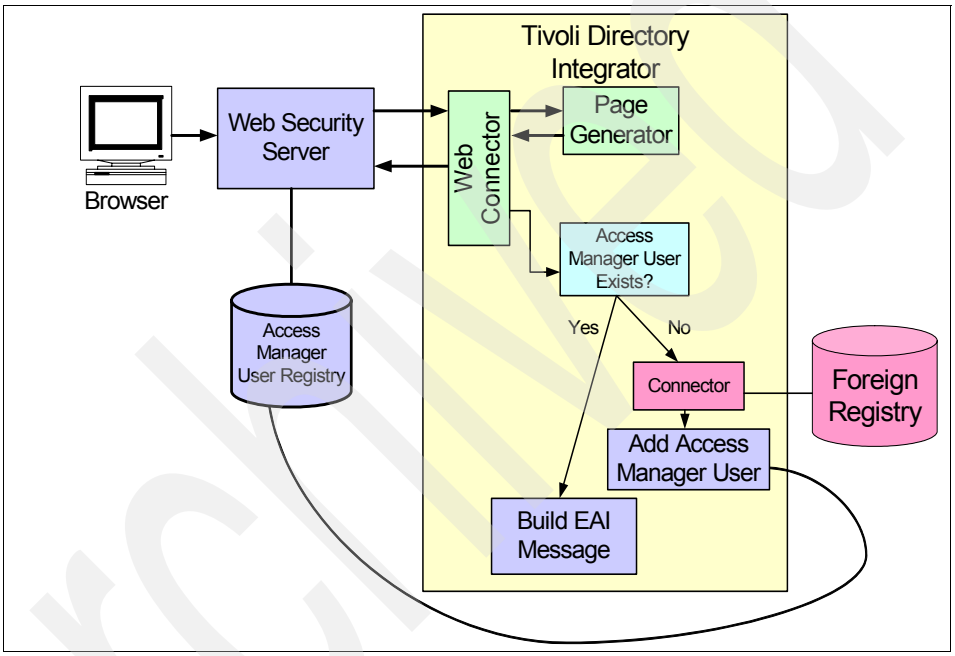


Figure 9-11 Automated user migration using EAI and IBM Tivoli Directory Integrator

9.4.7 Custom authentication using the External Authentication C API

All of the authentication mechanisms described above assume that the user identity validation information is held in the Access Manager user registry or can be verified locally on the Web security server. Of course, there are situations where this is not the case, and user authentication has to be performed outside of the Access Manager trusted domain: one-time password servers (for example, RSA SecureID), RADIUS, Resource Access Control Facility (RACF), and so on. On the other side, depending on the requirements, it may become necessary to extend or enrich the capabilities provided by built-in authentication libraries.

the Web security server provides a capability referred to as the *external authentication C API* in order to meet these requirements.

As shown in Figure 9-12, the external authentication C API enables you to substitute the default built-in Web security server authentication mechanism with a highly flexible shared library mechanism that allows custom handling and processing of client authentication information.

You can customize the external authentication C API shared library to handle authentication data according to your security requirements given the following options.

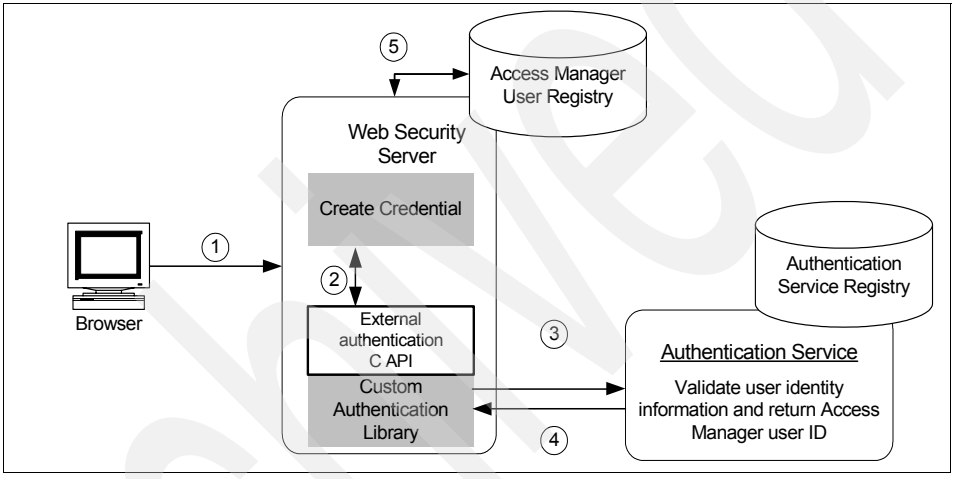


Figure 9-12 Web security server authentication model with the external authentication C API

The external authentication C API can process authentication data internally and return an Access Manager identity. This is especially useful if you want to have enriched authentication mechanisms in comparison to built-in ones, for example, checking client certificate validity via Online Certificate Status Protocol (OCSP). The user identity validation information may reside in the user registry and not be used for authentication by default (for example, providing a customer number along with user ID and password in B2B scenarios).

Extending the built-in capabilities of authentication mechanisms provided by Access Manager is another reason to built a custom program using the external authentication C API. This method enables you to authenticate clients who are not direct members of the Access Manager security domain. In that case, the custom external authentication C API program can direct authentication data to be processed by an external authentication mechanism and third-party registry (for example, RACF, One-Time Password Server, or authentication via personal

question). Ultimately, the external authentication application returns an Access Manager identity to the Web security server for creating the Access Manager credential.

9.4.8 Entitlement service interface

An entitlement service interface is a part of the aznAPI that is called during the building of a credential. This entitlement service receives the basic user credential being created and can specify a list of additional custom attributes to be added to the credential before it is returned to the application.

The entitlement service interface is called from within the aznAPI and so the function is available to all Access Manager applications regardless of the registry and regardless of the authentication method used. The credential attribute service can obtain the custom credentials from any source; they don't have to come from the user registry. Custom entitlement services can be written to obtain attributes from any desired source.

Figure 9-13 shows the architecture for adding attributes to a new user credential. The main aspect is that the Resource Manager can be any Access Manager aznAPI application—it is no longer limited to just the Web security servers.

Initially the application calls the aznAPI to request a credential. The aznAPI builds a basic Access Manager credential for the user (1) and then calls the configured *credential attribute* entitlement services. These gather additional attributes for the user (from the registry in this example) and return them to the aznAPI (2). The aznAPI then adds these attributes to the basic Access Manager credential before returning it to the calling application.

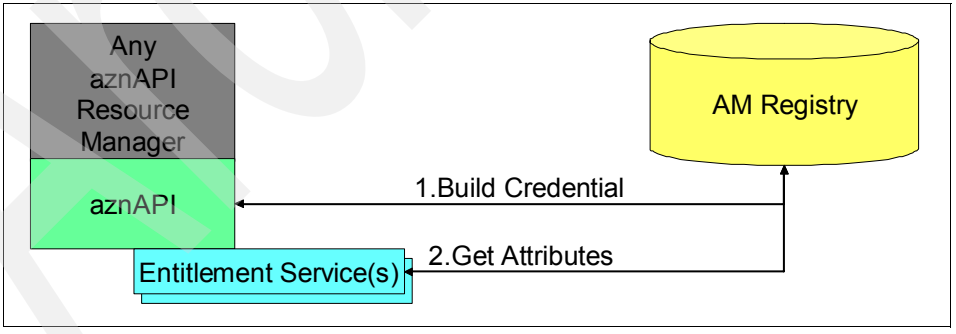


Figure 9-13 Entitlement service

An entitlement service is a very generic plug-in that can be called by the Access Manager authorization service. It is possible to register multiple credential

attribute entitlement services with the aznAPI. These will all be called, and all of the attributes are added to the user's credential.

The input to an entitlement service is a user credential and an application context. The output of an entitlement service is an attribute list. This is how the entitlement service passes back its results.

Credential attribute entitlement service

The credential attribute entitlement service extracts information from a user's LDAP entry and add its to their credential. For example, a backend application requires a user's department number in addition to their user ID in order to build the application interface appropriately. By using the credential attribute entitlement service, the Web security server can pull the user's department out of their entry in LDAP, place it in the user's credential, then use the information from the credential to place the department value in an HTTP header. These attributes must exist within the inetorgperson (or some subclass of) LDAP object.

Policy credential attribute entitlement service

The policy credential attribute entitlement service allows user policy information to be gathered from the Access Manager registry and inserted into the user's credential. This information can include password requirements, account expiration date, and time-of-day login restrictions. If a user does not have these attributes explicitly defined to their account, the values are inherited from the global policy settings.

9.4.9 Authentication using customized HTTP headers

Access Manager supports authentication via customized HTTP header information supplied by the client or a proxy agent.

This mechanism requires a mapping function (a shared library) that maps the trusted (pre-authenticated) header data to an Access Manager identity. The Web security server can take this identity and create a credential for the user.

The Web security server assumes that custom HTTP header data was authenticated previously. For this reason, you should implement this method exclusively with no other authentication methods enabled. It is possible to impersonate custom HTTP header data. This method is therefore only appropriate in tightly controlled networks, where traffic from the authenticating proxy supplying the HTTP header can be absolutely trusted.

By default, this shared library is built to map data from trusted proxy headers.

9.4.10 Authentication based on IP address

Access Manager supports authentication via an IP address supplied by the client.

This mechanism is used best in combination with other mechanisms. For example, you can use IP network addresses to identify a certain group of users, give them access to a certain application, then use additional authentication mechanisms to give access to more protected applications.

Such a configuration can be used to implement a two-factor authentication as well. It will possibly be more secure than plain password authentication.

9.4.11 No authentication

Any user who can reach the Web security server belongs to the group of unauthenticated users. This group can also get certain permissions.

This group of unauthenticated users generally is used to define public Web access. The Web security server can force unauthenticated users to use another authentication method when selecting certain protected URLs.

All users who can reach the Web security server might already have enough permissions to contact certain junctioned Web servers. For example, if the Web security server is connected to a VPN gateway, only authorized VPN users will be able to reach that server, and additional authentication might not be needed. In this situation, you can probably treat unauthenticated users as you would a group of password-authenticated Internet users.

9.4.12 MPA authentication

Access Manager provides an authentication mechanism for clients using a Multiplexing Proxy Agent (MPA). This is a special variation of the authentication with customized HTTP headers that is often used for mobile phones and PDAs, but is not limited to these.

Multiplexing Proxy Agents are gateways that accommodate multiple client access. IBM Everyplace Wireless Gateway (EWG) is an integrated part of the IBM WebSphere Everyplace Suite that provides security-rich wired and wireless connectivity between the IT network and the communications network; for example:

- Cellular networks, including GSM, CDMA, TDMA, PDC, PHS, iDEN, and AMPS
- Packet radio networks, including GPRS, CDPD, DatatTAC, and Mobitex

- Satellite and wireline environments, including DSL, cable modems, Internet service providers, ISDN, dial, and LAN

In addition, the Everyplace Wireless Gateway provides protocol translation as a Wireless Application Protocol (WAP) gateway, information push as a WAP push proxy gateway, and support for short messaging services (SMS). EWG establishes a single SSL channel to the origin server and “tunnels” all client requests and responses through this channel.

To the Web security server, the information across this channel initially appears as multiple requests from one client. The Web security server must distinguish between the authentication of the MPA server over SSL and the additional authentication requests for each individual client, so MPAs must use a different authentication method from that used by clients.

Because the Web security server maintains an SSL session state for the MPA, it cannot use SSL session IDs for each client simultaneously. The Web security server must therefore use some other mechanism for maintaining sessions with the clients, such as cookies or HTTP headers.

WebSEAL has support for the Entrust Proxy and the Nokia WAP gateway.

9.5 Web security server single sign-on mechanisms

After a user is authenticated by the Web security server and an authorization decision is made, the Web security server has to forward the user’s request to a back-end Web application server. If needed, the Web security server can include information about the user, such as X.509 distinguished name, group memberships, or any other value.

The mechanisms to forward that information can vary. You can use standard protocols such as the HTTP basic authentication header or use proprietary mechanisms when talking to specific server products. The Web security server supports several mechanisms for forwarding requests to Web application servers.

This section presents alternatives on how to pass information about the user and the user’s request to the back-end application.

When a protected resource is located on a junctioned Web application server, a client requesting that resource can be required to perform multiple logins: one for the Web security server server and one for the back-end server. Each login may require different login identities. Often, the problem of administering and maintaining multiple login identities can be solved with a single sign-on mechanism.

The Open Group defines single sign-on as a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where that user has access permission, without the need to enter multiple passwords[1]. While Tivoli Global Sign-On addresses the authentication issues for various applications running on different operating systems, the Web security server's realm is to provide the single sign-on functionalities for Web infrastructures. WebSEAL, acting as a Web reverse proxy to the company's Web environment, communicates with the junctioned servers on behalf of the users. It enables the user to access a resource, regardless of the resource's location, using only one initial login. Any more login requirements from back-end application servers are handled so that they are transparent to the user.

Depending on integration requirements, different data should be sent to the Web security server-secured Web application using different formats. However, most of the Web applications support standard HTTP-based mechanisms for the user identification, which are exploited by the Web security server.

9.5.1 Tivoli Global Single Sign-On lockbox

Most Web applications support basic authentication or forms-based login for checking authenticity and obtaining a user's identity information. When using this support, an application or the server the application is running on maintains a database with user IDs and passwords (in the most simple case). In our initial example in Chapter 7, "A basic WebSEAL scenario" on page 245, it was operating system-based user management on multiple Web servers, containing lists of user IDs and passwords. After challenging a user and obtaining a user ID and password, an application would look up the matching entry and, if one was found, the user was considered authenticated and his or her identity was associated with the provided user ID. In more sophisticated environments relational databases, existing applications or LDAP-based repositories are targeting that scope.

Access Manager supports a flexible single sign-on solution that features the ability to provide alternative user IDs and passwords to the Web application servers in two different ways:

- By supplying user ID and password information via basic authentication headers
- By performing forms-based single sign-on

The integration is achieved by creating SSO-aware junctions between the Web security server and Web servers hosting the applications. Global Sign-On (GSO) resources and GSO resource groups must first be created in Access Manager for

[1] From the security section of the Open Group Web site (http://www.opengroup.org/security/sso/).

every application that requires a different logon. When the Web security server receives a request for a resource located on the SSO-junctioned server, the Web security server queries the Access Manager user registry for the appropriate authentication information. The user registry contains mappings for each user registered for using that application, which provides alternative user IDs and passwords for specific resources. Evidently, that information has to be in the repository prior to initial using. The values (user IDs and passwords) should match those stored in the application home registry.

Note: Although junctions are set up on a Web server basis, it is possible to provide different SSO data to different applications hosted on the same server. In order to achieve this, multiple GSO junctions to the same Web server are created. However, using access control lists, the access to the resources is defined that way, so that only appropriate URLs can be requested through a specified junction.

The visible advantage of the solution is that no changes are supposed to be made on the application side. However, synchronization of the user IDs and passwords in the application's home user registry and Access Manager user registry is required and can be accomplished with IBM Tivoli Directory Integrator.

A special situation emerges if Access Manager and the secured application share the same repository for storing user data, as shown in Figure 9-14 on page 309. An LDAP directory is the most suitable platform for maintaining application-specific information about users and groups. Given compatible LDAP schemas, many applications may share the same LDAP directory. LDAP provides a standardized way of authenticating users based on user ID and password stored as user attributes. However, it provides no flexibility in defining object classes to be used for authenticating a user rather than performing a call based on primary identification attributes of a user (user ID and password).

While using an Access Manager GSO junction, Access Manager uses specific LDAP attributes for storing GSO information for every GSO user. As a result, the GSO user ID and password provided for a specific junction are not necessarily the same as the primary ones. However, a junctioned application sharing the same LDAP repository would then try to authenticate a user using these values against primary ones (by doing LDAP bind or compare). The need arises to keep the values of primary user IDs and passwords the same as GSO IDs and passwords.

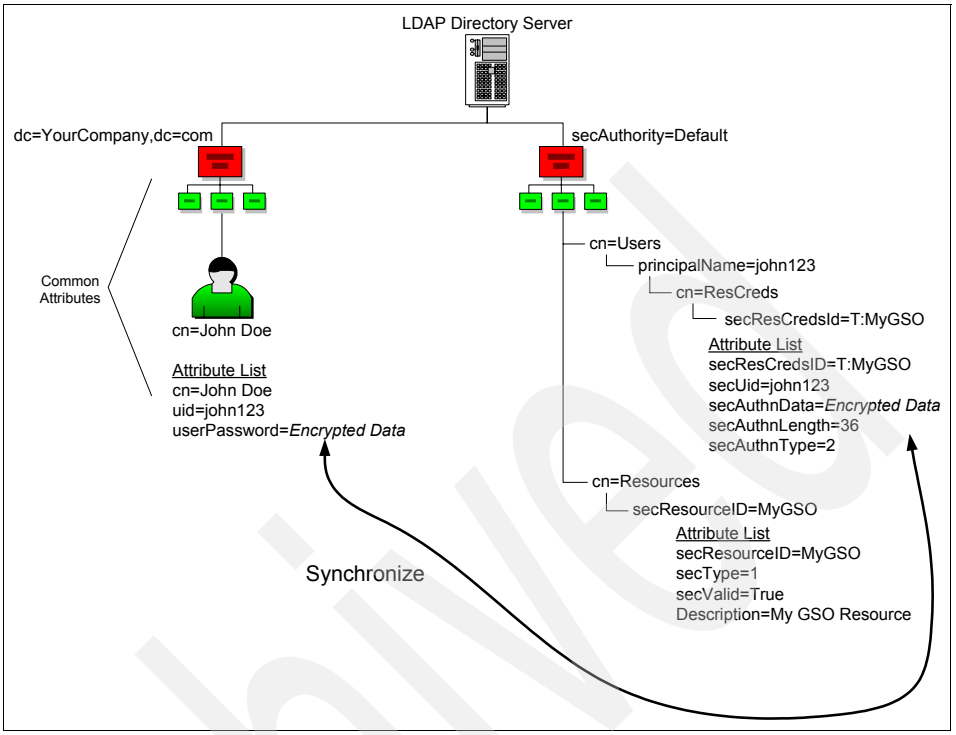


Figure 9-14 LDAP shared by Access Manager and other applications

The following issues should be considered while looking for solutions for integrating Access Manager and Web applications using the same LDAP repository or even different user repositories:

- Using a directory synchronization product such as IBM Tivoli Directory Integrator to synchronize both the corporate tree and the Access Manager tree within the same directory. Directory Integrator also allows for the synchronization of a user's password.
- As GSO passwords are encrypted, they can only be read by the Access Manager GSO APIs.

9.5.2 Forms single sign-on

Forms single sign-on authentication supports existing applications that use HTML forms for authentication that cannot be modified to directly trust the authentication performed by the Web security server. Forms single sign-on is built on the following process:

1. The Web security server will interrupt the authentication process initiated by the backend application.
2. The Web security server supplies the data required by the login form and submits the login on behalf of the user.
3. The Web security server saves and restores all cookies and headers.
4. User is unaware of the second login taking place between the Web security server and backend application.
5. The backend application is unaware that the login form is not coming directly from the user.

The login form from the backend application can be filled in with a variety of information from the Web security server such as the following:

- Static text.
- GSO user name and password (see “Tivoli Global Single Sign-On lockbox” on page 307 for more information).
- Values contained within a user’s credential.

In order to use forms single sign-on, the backend application’s login page must be uniquely identifiable. Also, client-side scripting can be used to validate input data, but it must not modify the input data. The junction where the authentication request is directed must be the same junction where the login page is returned.

9.5.3 Passing an unchanged basic authentication header

WebSEAL can be configured to pass the received basic authentication data unchanged to the junctioned application. If Access Manager and the application share the same LDAP registry, Access Manager authenticates a user against the same LDAP attributes as an application performing a regular LDAP bind (that is, using a main user ID and password). In this case, there is no need to maintain the GSO attributes of a user, and the main password may be encrypted. However, basic authentication is the only available authentication method used by WebSEAL, as WebSEAL has to obtain the BA header values in order to pass them through.

9.5.4 Providing a generic password

At this point, the following sections are based on the assumption that trust is established between WebSEAL and the back-end application server.

Given a Web application that may be contacted only through WebSEAL, an integration solution based on providing a user ID along with a uniform generic password and shared by WebSEAL and the application can be considered. As the process of authenticating a user is performed by WebSEAL, and given that WebSEAL is the only gateway into the application, there is no need to carry out the authenticity check again. Although no changes have to be made in the application, it still could perform authentication in its obvious manner. However, its scope should only be the gaining of user identity. There should be no other possibilities available to contact the application avoiding WebSEAL.

The application can maintain its own user repository or share that of Access Manager (LDAP-based). In the second case, however, the LDAP-bind issue discussed previously (see 9.5.1, “Tivoli Global Single Sign-On lockbox” on page 307) has to be considered. That leads to the necessity of maintaining separate entries for a single user for Access Manager and the secured application.

9.5.5 Supplying user and group information

The Web security server can be configured to provide information to a junctioned application about user ID, groups, and resources the user has access to. That is accomplished by supplying the values of defined HTTP variables:

iv_user	For user ID
iv_user_l	For user’s LDAP distinguished name
iv_groups	For groups a particular user belongs to
iv_creds	For the user’s credentials in base64-encoded Privilege Attribute Certificate (PAC) format

The variables supplied in the HTTP stream can be mapped easily to the CGI environment variables that can be interpreted by a Web application. As no password information can be supplied this way, no authentication can be performed by the junctioned Web application. However, it is possible to combine this option with any previously described.

Secure credential exchange

We briefly introduce the notion of secure credentials and how they could be exchanged between Web applications.

Credentials are created as a result of a successful authentication. Credentials created by the Web security server can be understood by other Web security servers in the same Access Manager security domain and even beyond. However, the credential exchange with the junctioned Web server is not necessarily trivial mainly due to the lack of standardization.

9.5.6 Using LTPA authentication with the Web security servers

The Web security server can provide authentication and authorization services and protection to an IBM WebSphere or Lotus Domino environment. When the Web security server is positioned as a protective front end to WebSphere or Lotus Domino, accessing clients are faced with two potential login points. Therefore, the Web security server supports a single sign-on solution to one or more IBM WebSphere or Lotus Domino.

WebSphere provides the cookie-based *Lightweight Third Party Authentication* mechanism (LTPA). You can configure the Web security server to support LTPA and provide a single sign-on solution for clients.

When a user makes a request for a WebSphere or Lotus Domino resource, the user must first authenticate to the Web security server. Upon successful authentication, the Web security server generates an LTPA cookie on behalf of the user. The LTPA cookie, which serves as an authentication token for WebSphere or Lotus Domino, contains user identity and password information. This information is encrypted using a password-protected secret key shared between the Web security server and the WebSphere or Lotus Domino server.

The Web security server inserts the cookie into the HTTP header of the request that is sent across the junction to WebSphere or Lotus Domino. The back-end WebSphere or Lotus Domino server receives the request, decrypts the cookie, and authenticates the user based on the identity information supplied in the cookie.

To improve performance, the Web security server can store the LTPA cookie in a cache and use the cached LTPA cookie for subsequent requests during the same user session. You can configure lifetime timeout and idle (inactivity) timeout values for the cached cookie.

The creation, encryption, and decryption of LTPA cookies basically introduces processing overhead. The LTPA cache functionality enables you to improve the performance of LTPA junctions in a high load environment. By default, the LTPA cache is enabled. Without the enhancement of the cache, a new LTPA cookie is created and encrypted for each subsequent user request.

Having the LTPA cookie enabled is independent of the basic authentication header. This means that with the LTPA cookie inserted into the request header, it is still possible to have the BA header to carry any authentication information to the back-end server depending on the -b option specified during the junction creation. The usage of the BA header depends on the configuration of the back-end WebSphere or Lotus Domino server. Figure 9-15 shows the available usage scenarios with the LTPA authentication.

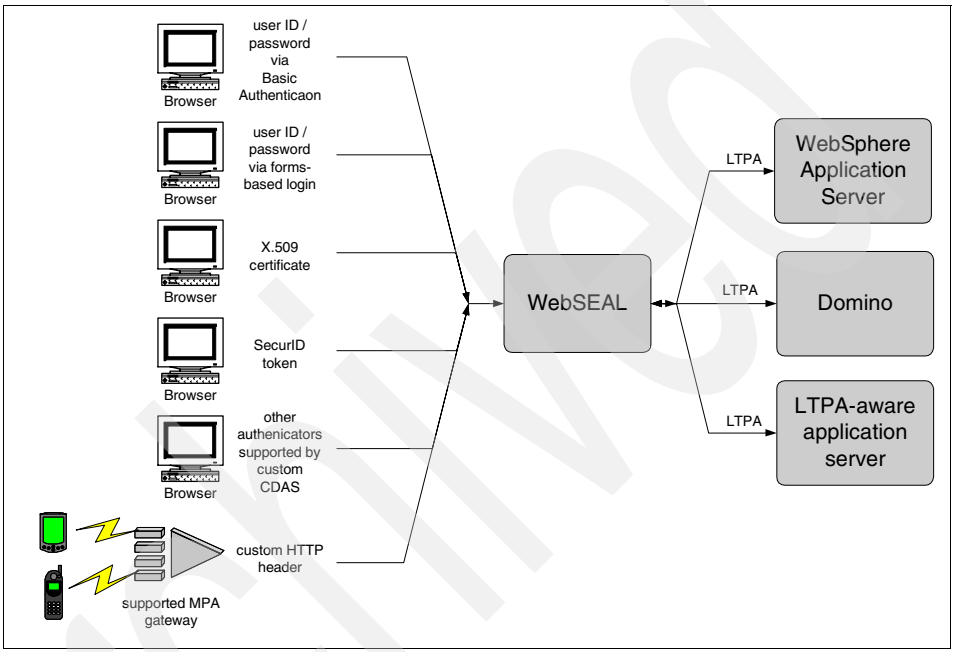


Figure 9-15 WebSEAL LTPA token single sign-on

This concludes the discussion on the various authentication capabilities provided by Tivoli Access Manager. In the next chapter we take a closer look into the authorization realm.

9.6 Enterprise single sign-on mechanisms

Another type of distributed scenario involves single sign-on for access to resources in multiple security domains. A single domain ties together a group of individuals who need access to a set of applications within an organization. This implies that if a user needs to access an application in a different security domain, then they would need to perform a single sign-on operation to that domain in order to gain access. When secured by Tivoli Access Manager for e-business, this implies a separate policy server and user registry.

The term multiple security domains can also refer to a multi-domain Access Manager environment where the user registry and Policy Server are shared, but the security policy is separated into different authorization databases on a per-domain basis. For more information about multi-domain environments, see the product feature outlined in the section titled “Multi-Domain Policy Server” on page 179.

Consider two divisions of a company, each offering Web-based services to Internet customers. Each division deployed Web security servers in separate security domains, implying that there is a separate Policy Server and user registry for each domain. However, there is a requirement for certain users in one domain to access resources in the other domain without needing to authenticate twice. The Web security servers support two different types of cross-domain authentication to address such scenarios: Cross Domain Single Sign-On and e-community single sign-on. For both of these types of single sign-on mechanisms to work, it is necessary that the users participating in the single sign-on can be mapped to a user in the opposing security domain. Typically, this might be a one-to-one mapping; however, a programatic interface is provided (cross domain mapping function) if the customer wants to implement a more complex mapping function.

Note: With the release of Tivoli Federated Identity Manager, many customers are embracing the open standards approach to enterprise single sign-on by adopting a ratified standard for passing security tokens between partners. More information can be found in Part 4, “Managing federations” on page 653.

9.6.1 Cross Domain Single Sign-On

The Web security servers support the ability to forward an authenticated identity from a user in one security domain to a Web security server in another security domain. The *receiving* Web security server then maps the identity provided by the *sending* Web security server to an identity that is valid in its security domain. This functionality can also be viewed as a *push* model with respect to authentication.

This functionality is known as Cross Domain Single Sign-On (CDSSO). In CDSSO, the user makes a request to a special link on a Web security server, which then forwards the request, along with some encrypted user and session information to a Web security server in a different Access Manager domain. The destination Web security server recognizes the incoming request as a CDSSO request and authenticates the incoming encrypted token within the local domain. Hence each security domain shares a set of crpyographic keys.

Note: Cross Domain Single Sign-On requires that the originating site's application be able to generate an appropriate CDSSO link for the destination site, or alternatively, the link is housed on the static content near the logout button.

If these changes are not permitted/desired or if application awareness of Web security server single sign-on functionality is not possible, then e-community single sign-on can be used. See 9.6.2, "e-community single sign-on" on page 316 for more information.

The CDSSO process contains the following steps:

1. A user initially logs on to a Web security server in one security domain.
2. At some point the user accesses a link controlled by the user's Web security server, which contains a special directive (pkmscdsso). This directive results in redirecting the user to a URL controlled by a Web security server in another security domain and passing encrypted credential information to the new Web security server.
3. The user is redirected to the other Web security server and this server decrypts the credential information passed to it, maps the identity to one defined in its own user registry, and creates a local credential. The Web security server then associates this credential with a local session.
4. At this point the user has established secure sessions with two Web security servers in different domains, but has only had to log in once.

Figure 9-16 on page 316 summarizes a typical CDSSO flow.

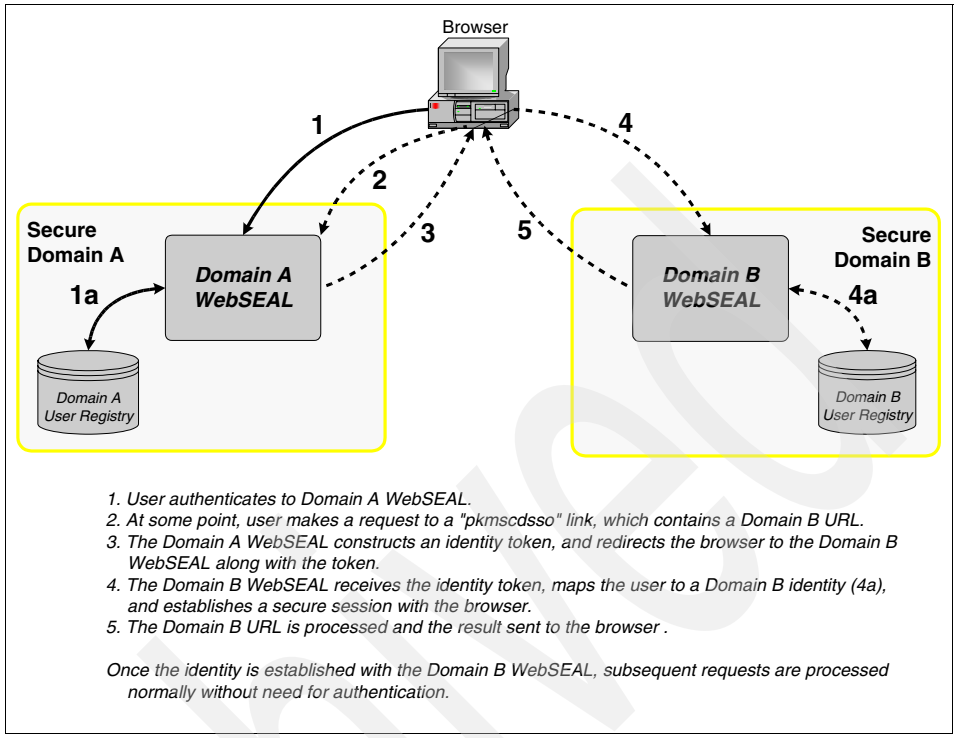


Figure 9-16 CDSSO identity determination process

Another significant CDSSO implication for a given security domain, in addition to the need to potentially modify backend applications, involves the mapping of user identities. How this mapping is done is not really an architectural issue. It is more a detailed-design/implementation concern.

It is possible (using the CDMF interfaces discussed in 9.6.3, “Cross Domain Mapping Framework” on page 321) to map from an ID in one domain to a different ID in another. However, if the IDs in both domains are the same, a direct mapping may be done. This is the default and does not require the use of any special programming interfaces.

9.6.2 e-community single sign-on

e-community single sign-on supports a cross-domain authentication capability. However, it differs from CDSSO in a few ways. Recall that in CDSSO, authenticated identities are *forwarded*. In an e-community scenario, identities are instead retrieved—it is a *pull* model. The use of e-communities has certain advantages over CDSSO, yet have architectural impacts that are not encountered in a CDSSO environment.

Instead of having to use special URLs to indicate the use of single sign-on as in the CDSSO model, e-community allows for direct access to secured links. This has a benefit over CDSSO in that users can bookmark links to resources but will still be allowed to participate in e-community.

In this model, multiple Access Manager domains are defined to be part of a single e-community. While each participating domain has its own user registry, one of the domains is designated to be the *home domain*. Users requesting protected resources in any of the participating domains initially authenticate to a *Master Authentication Server (MAS)* in the home domain. After the initial authentication occurs, the user has an e-community identity based on the home domain's user registry. A user's e-community identity subsequently may be mapped, as required, to local identities by Web security servers in other domains within the e-community.

Note: Users who do not exist in the MAS home domain can still authenticate to their own domain and access protected resources. This allows for a company to restrict access to who can essentially single sign-on to resources. Only users defined to both the MAS domain and the domain where the protected resource is defined can participate in e-community single sign-on.

Also, the MAS architecture can also be deployed in a single Access Manager domain. This allows for organizations to deploy a central authentication server, allowing other servers to simply provide content delivery.

The e-community model is shown in Figure 9-17 on page 318. Following are some key points to be aware of in the e-community model:

- There is a single-home domain for the entire e-community. All users will authenticate to this domain first.
- The MAS belongs to the home domain and should not be used for any other purpose than to authenticate users. There should not be any authorization performed by the MAS.
- After authentication by the MAS, the home domain identity provided in the e-community token is mapped to a user identity in the domain of the referring WebSEAL.
- The MAS also has the ability to retrieve information from a user's credential in the home domain and place it in the e-community token on a per domain basis. This allows the MAS to tailor the token contents to match the needs of the destination domain.

Single sign-on with e-community can be used if there are two completely separate Access Manager security environments (two different Policy Servers and user registries) or in an Access Manager multi-domain environment where

there is one Policy Server and one user registry shared between domains (this does not imply that the users and groups are shared between domains though). Refer to “Multi-Domain Policy Server” on page 179 for more information about Access Manager multiple domain environments with a single Policy Server.

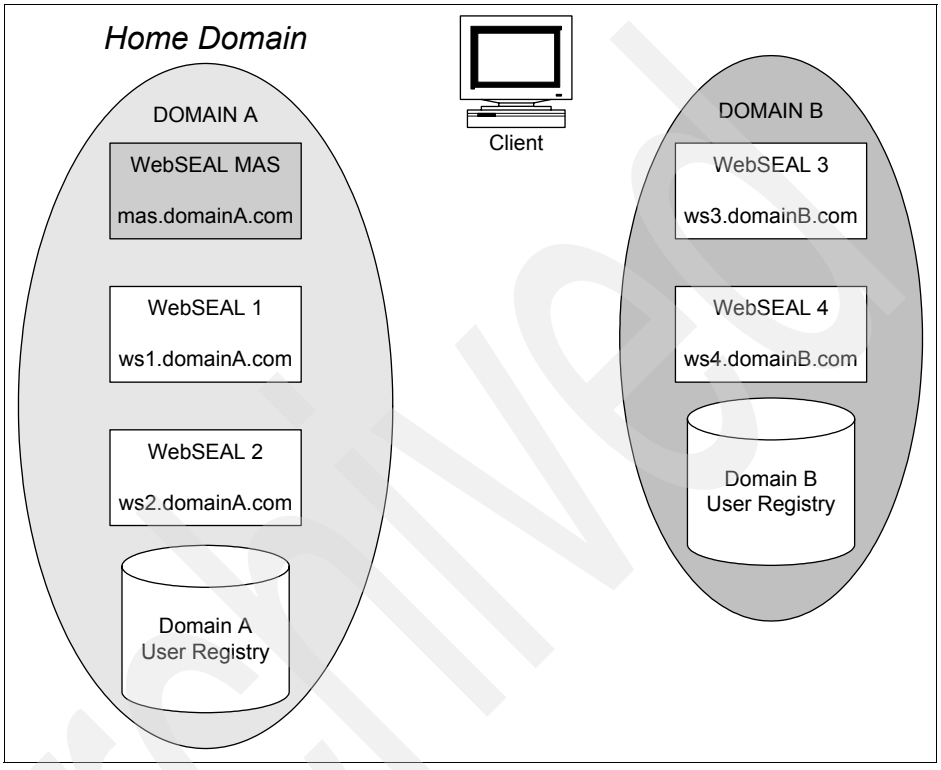


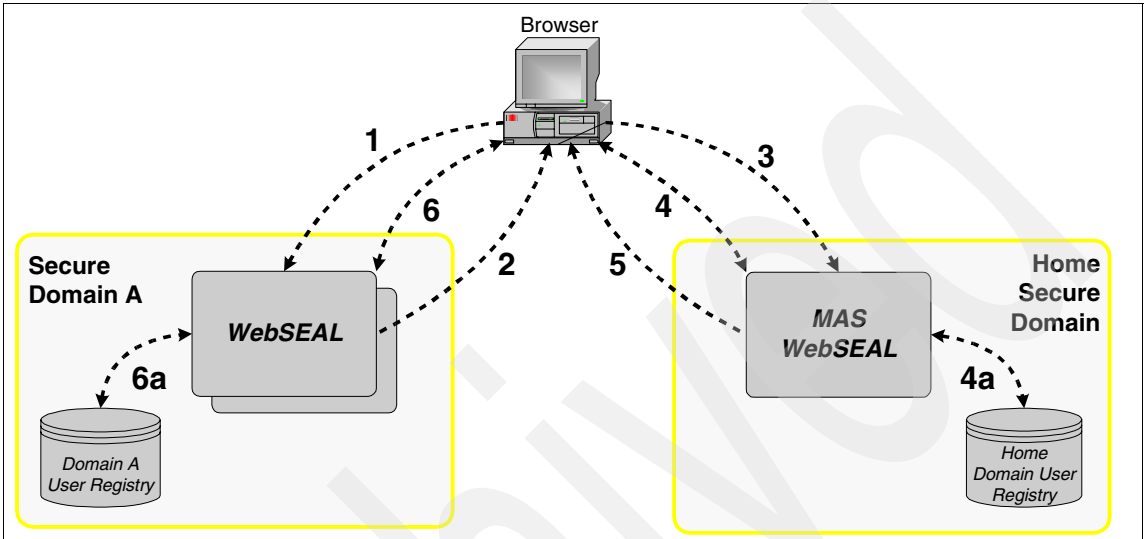
Figure 9-17 e-community single sign-on model

The e-community mechanism involves the following steps, we use WebSEAL as the example of the Web security server configured:

1. A user makes a request for a protected resource controlled by a WebSEAL server in one of the e-community domains. This WebSEAL does not yet have an established secure session with this user.
2. The WebSEAL server redirects the user to the MAS and sends with the request a special directive (pkmsvouchfor), which requests that the MAS provide identity information for the user.
3. The MAS checks to see whether the user has already been authenticated to the e-community, and if not, the MAS then authenticates the user.
4. The MAS then sends a token back to the original WebSEAL server that contains credential information that vouches for the user’s identity.

5. The WebSEAL server then maps the identity provided to it by the MAS to an appropriate Access Manager within its local domain and establishes a secure session with the browser.

Figure 9-18 summarizes the flow of an initial e-community user authentication.

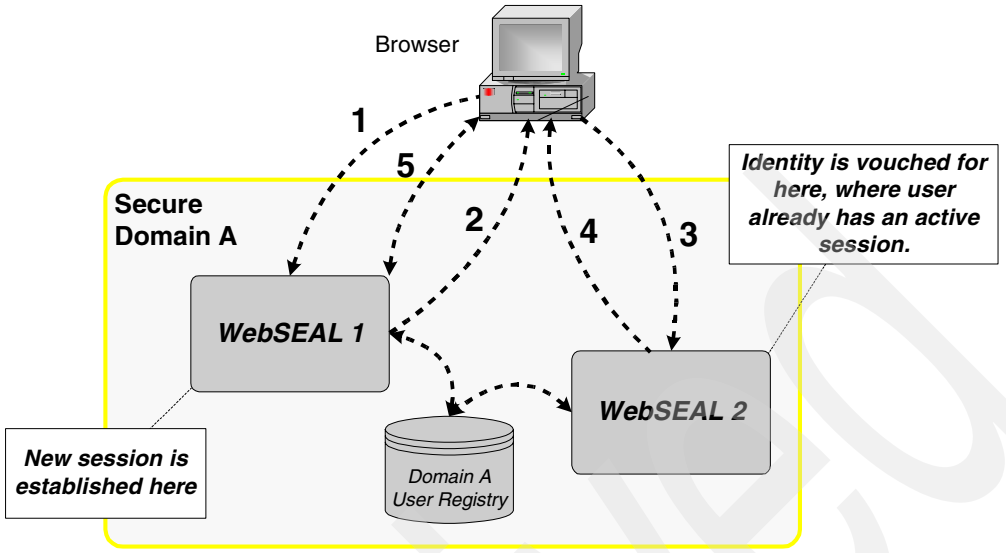


1. User requests URL:
   https://www.xyz.com/abc.html
2. WebSEAL redirects user to MAS WebSEAL with request for "voucher".
3. Browser forwards voucher request to MAS.
4. If a session has not yet been established, the user is authenticated and mapped to a Home Domain identity (4a).
5. A voucher token is created (in a cookie), and the MAS redirects the user back to the original WebSEAL.
6. The browser forwards the request and voucher cookie to the original WebSEAL, which maps the user to an appropriate Domain A identity (6a), establishes a secure session, and processes the request.

Once the session is established with the Domain A WebSEAL, subsequent requests are processed normally without need for vouchering/authentication.

Figure 9-18 e-Community initial identity determination process

Within the *home domain*, unauthenticated requests are always vouched for via the MAS. In other participating domains, after the user initially logs in to the MAS, subsequent authentication activities to other WebSEAL servers in those domains are handled locally. The first WebSEAL in the requestor's local domain that validates the user's identity against the MAS acts as the voucher for that user's identity within that domain for all subsequent authentication attempts. This is depicted in Figure 9-19 on page 320.



(User has authenticated to an e-community MAS in a previous request to WebSEAL 2. WebSEAL 2 now will vouch for subsequent identity "voucher" requests by other WebSEALs for this user in this domain.)

1. User requests URL from WebSEAL 1:
https://www.xyz.com/abc.html
2. WebSEAL 1 redirects user to WebSEAL 2 with request for "voucher".
3. Browser forwards voucher request to WebSEAL 2.
4. WebSEAL 2 provides a voucher cookie token, and redirects the user back to WebSEAL 1.
5. The browser forwards the request and voucher cookie to WebSEAL 1, which maps the user to the correct Domain A identity, establishes a secure session, and processes the request.

Once the session is established with the WebSEAL 1, subsequent requests are processed normally without need for vouchering/authentication.

Figure 9-19 e-Community subsequent identity determination process

The key advantage of e-community single sign-on over CDSSO is that the initial URL request can be made directly to the target WebSEAL server. Recall that with CDSSO, the URL request must go through the WebSEAL to which the user is currently authenticated. In an e-community configuration, the target WebSEAL is specifically configured to *retrieve* credential information through the vouching mechanism, and the URL request itself need not be accompanied by special processing or contain special characteristics, as in the CDSSO case. This can lead to scalability related issues that need to be addressed around the authentication event.

Architectural significance of e-community single sign-on

There are many design considerations regarding the implementation of e-community single sign-on, which will not affect your physical architecture design. The main architectural impact of e-community single sign-on involves the role of the MAS as the single authentication point. The key issue is that with all user authentication for the e-community going through a single domain, where should the MAS server (or servers) be located?

9.6.3 Cross Domain Mapping Framework

Whenever single sign-on is used between two different security domains, the need for mapping user IDs from one domain to another exists. The reason this issue occurs is that the user ID from one domain may not map exactly to a user ID in another domain. For example, a user may have an ID of user123 in one Access Manager security domain.The same user may have an ID of myuser123 in another Access Manager domain.

The Cross Domain Mapping Framework (CDMF) is a programming interface that may be used in conjunction with WebSEAL e-community single sign-on and Cross Domain Single Sign-On. It enables a developer to customize the mapping of user identities and the handling of user attributes when single sign-on functions are used.

Conceptually, the mapping in a CDMF function works in a manner similar to an application using the external authentication C API, except that it is used to map an Access Manager user in one secure domain to an Access Manager user defined in a different secure domain.

9.6.4 Cookie Based single sign-on

A number of customers have embraced the use of the Web security server failover cookie as a means for single sign-on. This allows Web security servers hosted in the same DNS domain to accept failover cookies configured from other domains. Of course, for this to function, failover cookies need to be configured as domain cookies and both hosted servers must share the cryptographic key.

With the addition of the Session Management Server, customer's now have the ability to perform Single Sign-on across Web security servers in the same domain configured into the Session Management Server. The advantage of this approach is that, unlike failover cookies, Session Management Server cookies are not cryptographic cookies containing user information.

10

Access Manager authorization

This chapter discusses the authorization mechanisms and the components of the Tivoli Access Manager authorization service, what they deliver, and how to apply them in the definition and enforcement of a comprehensive security policy. These include:

- Access control list (ACL)
- Protected object policy (POP)
- Authorization rule
- Object space
- Resource manager

The chapter uses several scenarios to illustrate where and how to apply the authorization components in cases ranging from simple static Web page access control to complex dynamic access control decisions.

10.1 Authorization overview

ISO 7498-2[1], the ISO Security Architecture, defines access control as:

> *The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.*

Tivoli Access Manager's core function is to support this definition. The Access Manager access control capabilities are built on a standards-based approach, namely the Open Group's authorization (azn) API standard. This technical standard defines a generic application programming interface for access control in systems whose access control facilities conform to the architectural framework described in International Standard ISO 10181-3[2] (access control framework).

The ISO 10181-3 framework defines four roles for components participating in an access request, similar to Figure 10-1 on page 325:

Initiator Initiators submit access requests. This request specifies an operation to be performed on a target.

Target A target can be an information or a system resource.

Access Control Enforcement Functions (AEFs)
AEFs submit decision requests to Access Control Decision Functions (ADFs). A decision request asks whether a particular access request should be granted or denied.

Access Control Decision Functions (ADF)
ADFs decide whether access requests should be granted or denied based on security policy.

ADFs make access control decisions based on Access Control Decision Information (ADI) or, in Tivoli Access Manager terms, Security Policy. ADI describes security-relevant properties of the initiator, the target, the access request, and the system and its environment.

[1] For a complete reference of ISO 7498-2 visit:
http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=14256
[2] For a complete reference of ISO 10181-3 visit:
http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=18199

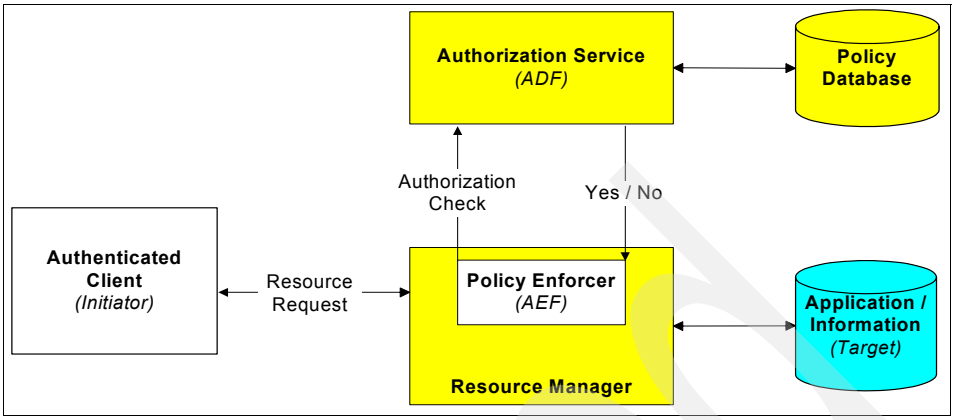


Figure 10-1 Open Group authorization model

10.1.1 The Tivoli Access Manager authorization service

The Tivoli Access Manager authorization service is responsible for the authorization decision-making process that helps to enforce a security policy. Authorization decisions made by the authorization service result in the approval or denial of client requests to perform operations on protected resources in a domain.

The authorization service is made up of three basic components:

- Master authorization policy database
- Policy Server
- The authorization decision-making evaluator

Policy database

The policy database, also referred to as the master authorization policy database and the master authorization database (ACL DB), contains the security policy information for all resources in a domain. Each domain has its own policy database. The content of this database is manipulated using the Web Portal Manager, the pdadmin command line utility, or the administration API.

Policy Server

The Policy Server (pdmgrd) maintains the policy database, replicates this policy information throughout the domains, and updates the database replicas whenever a change is made to the master.

The Policy Server also maintains location information about the other Tivoli Access Manager and non-Tivoli Access Manager resource managers operating in the domain.

Authorization evaluator

The authorization evaluator is the decision-making process that determines a client's ability to access a protected resource based on the security policy. The evaluator makes its recommendation to the resource manager which, in turn, responds accordingly.

Registry database replication parameters are configured for each evaluator.

Figure 10-2 illustrates the main components of the authorization service.

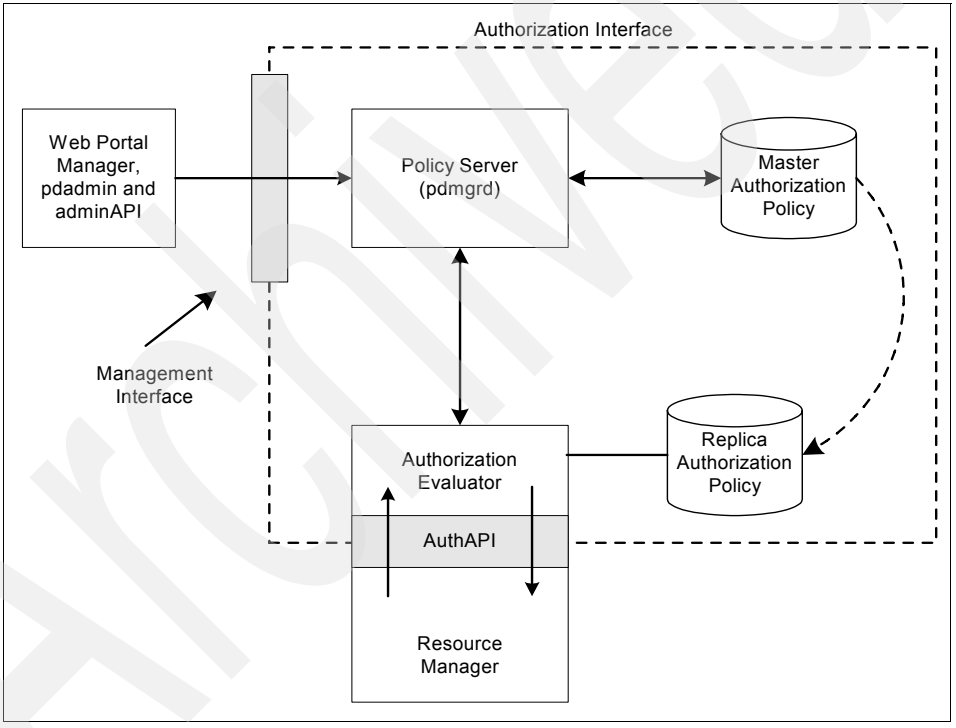


Figure 10-2 Access Manager authorization service

The diagrams in Figure 10-1 on page 325 and Figure 10-2 on page 326 clearly show how the Tivoli Access Manager authorization model maps to the ISO 10181-3 security architecture standard. Table 10-1 shows the mapping.

Table 10-1 Tivoli Access Manager authorization model mapping to ISO 10181-3

ISO 10181-3	Tivoli Access Manager
Initiator	User
Targets	Protected resources
AEF	Authorization Evaluator
ADF	Resource Manager
ADI	Policy Database

Authorization service interfaces

The authorization service has several interfaces where interaction takes place:

- Management interface: The security administrator manages the security policy by using the Web Portal Manager or the pdadmin command line utility to apply policy rules on resources in a domain. Both of these interfaces are built on the administration API. This API can also be used directly by applications to query and update management data.

  This interface requires detailed knowledge of the object space, policies, and credentials.

- Authorization API: The authorization API passes requests for authorization decisions from the resource manager to the authorization evaluator, which then passes back a recommendation whether the request should be granted or denied.

- JAAS: The Access Manager Authorization Java Classes provide an implementation of Java security code that is fully compliant with the Java 2 security model and the Java Authentication and Authorization Services (JAAS) extensions. This enables Access Manager to be used as an authentication and authorization back-end inside the Java 2 security model.

- .NET Authorization: The .NET Assembly for Tivoli Access Manager Authorization Services exposes Tivoli Access Manager APIs at the .NET Common Language Runtime Level, thereby making authorization functions available to all .NET languages.

Resource managers

An authorization service must be able to make appropriate access control decisions in the right place. That is, access control decisions must be enforced at the time they are required.

Resource managers support this objective. That is, they intercept application flow to request authorization decisions from the Authorization evaluator when a request is made for a protected object. Access Manager provides several resource managers, which are described in the Access Manager component overview in Chapter 5, “Access Manager core components” on page 163.

10.1.2 Access Manager authorization components

Within any Access Manager domain authorization enforces the security policy by determining what objects (targets) a user (initiator) can access and what actions a user can take on those objects, then granting appropriate access to the user.

Tivoli Access Manager handles authorization through the use of the following policy components (ADI):

- *Access control lists* (ACLs), *protected object policies* (POPs), and *authorization rules* for fine-grained access control
- Protected object space
- Users and groups

It uses the following enforcement mechanism (AEF):

- Resource managers

These use the following access control decision mechanisms (ADF):

- Tivoli Access Manager authorization service
- Standards-based authorization API, using the aznAPI for C language applications and the Java Authentication and Authorization Service (JAAS) for Java language applications
- External authorization service capability

The following sections describe these components and mechanisms and how they relate to each other.

10.2 Security policy

The goal of any security policy is to adequately protect business assets and resources with a minimal amount of administrative effort. First, you must define what resources should be protected. These could be any type of data object such as files, directories, network servers, messages, databases, or Web resources. Then, you must decide what users and groups of users should have access to these protected resources. You also need to decide what type of access should be permitted to these resources. Finally, you must apply the

proper security policy on these resources to ensure that only the right users can access them.

Access to objects within a domain is controlled by applying a security policy to the container and resource objects in the protected object space. Security policy can be explicitly applied to an object or inherited by the object from objects above it in the hierarchy. You need to apply an explicit security policy in the protected object space only at those points in the hierarchy where the rules must change.

Security policy is defined using a combination of:

- Access control lists (ACLs)

  An access control list specifies the predefined actions that a set of users and groups can perform on an object. For example, a specific set of groups or users can be granted read access to the object.

- Protected object policies (POPs)

  A protected object policy specifies access conditions associated with an object that affect all users and groups. For example, a time-of-day restriction can be placed on the object that excludes all users and groups from accessing the object during the specified time.

- Authorization rules

  An authorization rule specifies a complex condition that is evaluated to determine whether access will be permitted. The data used to make this decision can be based on the context of the request, the current environment, or other external factors. For example, a request to modify an object more than five times in an 8-hour period could be denied.

A security policy is implemented by strategically applying ACLs, POPs, and authorization rules to those resources requiring protection. The Tivoli Access Manager authorization service makes decisions to permit or deny access to resources based on the credentials of the user making the request and the specific permissions and conditions set in the ACLs, POPs, and authorization rules.

Authorization flow

Figure 10-3 on page 330 shows where ACLs, POPs, and authorization rules fall in the authorization process.

When an authorization decision request is received, the access control list for the object is checked first. If this does not allow access to the object then the request is denied. No further processing is required and no rule is evaluated.

If the ACL is satisfied, then the POP is checked. If the POP returns a *deny* decision (for example, if the time-of-day check fails), then the overall access is denied. No further process is required and no rule is evaluated.

If both the ACL and POP allow access, then the rules engine is called, and the engine's output ultimately determines whether access is permitted or denied.

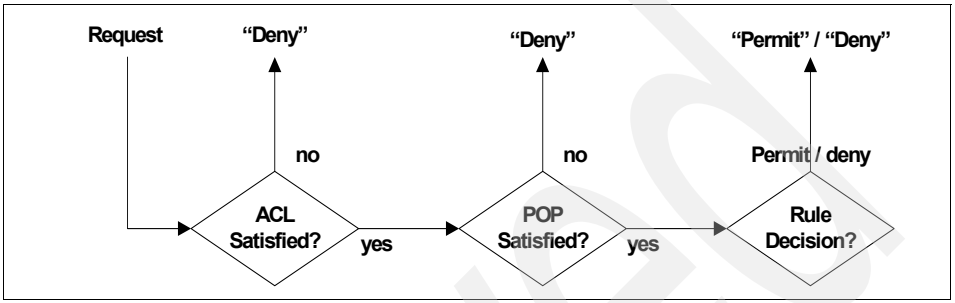


Figure 10-3 Authorization decision flow

The authorization engine uses the following algorithm to process the policy attached to a protected object:

1. Check ACL permissions. See "Evaluating an ACL" on page 333 for information about the ACL evaluation process.

   The ACL is also checked to determine whether the user (for whom the authorization check is being made) has the additional privilege of being unaffected by POP or authorization rule policy. This privilege is bestowed when the user's effective ACL for access to the object contains the B permission to denote that POP policy is ignored, or the R permission to denote that authorization rule policy is ignored.
2. When an authorization rule is attached to the object and the user does not have the privilege of being unaffected by authorization rules, verify that all of the ADI is present for the coming rule evaluation. If it is not, then find it by querying one of the available sources.
3. When there is a POP attached, check the Internet Protocol (IP) endpoint authentication method policy.
4. When there is a POP attached, check the time-of-day policy on the POP.
5. When there is a POP attached, check the audit-level policy on the POP, and audit the access decision as directed.
6. When an authorization rule is attached to the object and the user does not have the privilege of being unaffected by authorization rules, check the authorization rule policy.

7. When an external authorization service (EAS) operation or POP trigger applies to this access decision, invoke the external authorization services that apply.

If any of the ACL, POP, or authorization rule evaluations fail, then the access request is denied. The external authorization service can override this decision on its own, if it has been designed to do so, or it might choose not to participate in the authorization decision at all.

Every ACL, POP, or authorization rule can be thought of as a policy. Fill in the policy, specifying the appropriate access conditions. After the policy is complete, apply it to any number of resources within the domain. Subsequent changes to the policy are automatically reflected across the domain.

10.2.1 Protected object space

Tivoli Access Manager represents resources within a domain using a virtual representation called the protected object space. The protected object space is the logical and hierarchical portrayal of resources belonging to a domain.

The protected object space consists of two types of objects:

Resource objects Resource objects are the logical representation of actual physical resources, such as files, services, Web resources, message queues, and so on, in a domain.

Container objects Container objects are structural components that enable you to group resource objects hierarchically into distinct functional regions.

Security policy can be applied to both types of objects. Figure 10-4 on page 332 shows a logical representation of a protected object space with multiple container and resource objects.

The structural top, or start, of the protected object space is the root container object, which is represented by a forward slash (/) character. Below the root container object are one or more container objects. Each container object represents an object space consisting of a related set of resources. These resources can be resource objects or other container objects.

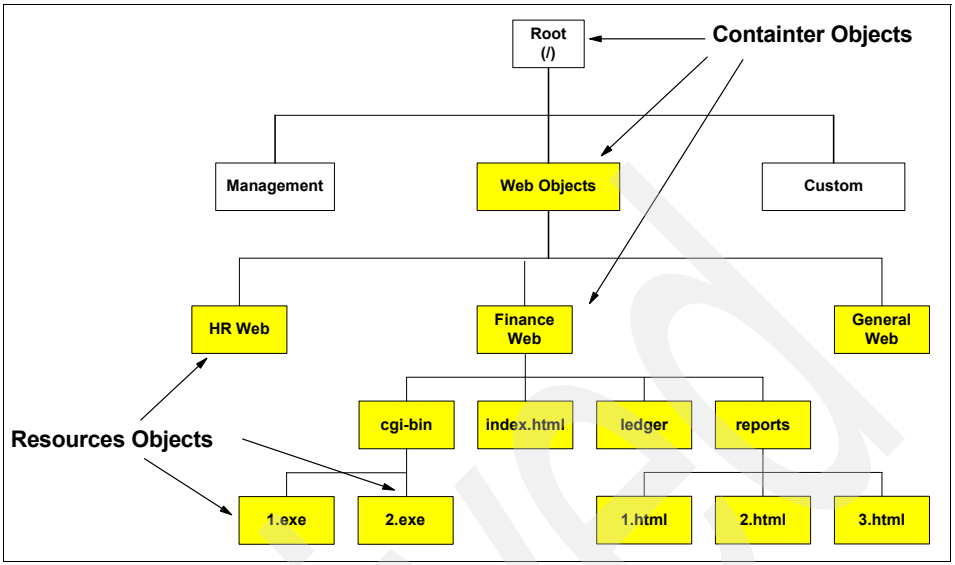


Figure 10-4 Access Manager protected object space

Tivoli Access Manager creates an object space called /Management that consists of the objects used to manage Tivoli Access Manager itself. Each resource manager that protects a related set of resources creates its own object space. For instance, the WebSEAL resource manager, which protects Web-based information and resources, creates an object space called /WebSEAL. These companion applications are referred to as blades.

10.2.2 Users and groups

Tivoli Access Manager maintains information about Tivoli Access Manager users and groups in the user registry. Users and groups that already exist in the user registry can be imported into Tivoli Access Manager. If a user or group does not already exist in the user registry, it can be created directly within Tivoli Access Manager.

When a user is authenticated to Tivoli Access Manager, a user credential is returned. This credential is used by other Tivoli Access Manager functions to uniquely identify the user making the request.

10.2.3 ACL policy

The policy that defines who has access to an object, and what operations can be performed on the object, is known as the ACL policy. Each ACL policy has a unique name and can be applied to multiple objects within a domain. An ACL policy consists of one or more entries describing:

- The names of users and groups whose access to the object is explicitly controlled
- The specific operations permitted to each user, group, or role
- The specific operations permitted to the special any-other and unauthenticated user categories

Using ACL policies with the authorization service

Tivoli Access Manager relies on ACL policies to specify the conditions necessary for a particular user to perform an operation on a protected object. When an ACL is attached to an object, entries in the ACL specify what operations are allowed on this object and who can perform those operations.

Resource manager software typically contains one or more operations that are performed on protected resources. Tivoli Access Manager requires these applications to make calls into the authorization service before the requested operation is allowed to progress. This call is made through the authorization application programming interface (authorization API) for both Tivoli Access Manager services and other applications.

The authorization service uses the information contained in the ACL to provide a simple yes or no response to the question: Does this *user* (group) have the *r* permission (for example) to *view* the requested *resource*?

The authorization service has no knowledge about the operation requiring the r permission. It is merely noting the presence, or not, of the r permission in the ACL entry of the requesting user or group. The authorization service is completely independent of the operations being requested. This is why it is easy to extend the benefits of the authorization service to other applications.

Evaluating an ACL

Tivoli Access Manager follows a specific evaluation process to determine the permissions granted to a particular user by an ACL. When you understand this process, you can determine how best to keep unwanted users from gaining access to resources.

Evaluating authenticated requests

Tivoli Access Manager evaluates an authenticated user request in the following order:

1. Match the user ID with the ACLs user entries. The permissions granted are those in the matching entry.

   Successful match: Evaluation stops here.
   Unsuccessful match: Continue to the next step.

2. Determine the groups to which the user belongs and match with the ACLs group entries: If more than one group entry is matched, the resulting permissions are a logical *or* (most permissive) of the permissions granted by each matching entry.

   Successful match: Evaluation stops here.
   Unsuccessful match: Continue to the next step.

3. Grant the permissions of the any-other entry (if it exists).

   Successful match: Evaluation stops here.
   Unsuccessful match: Continue to the next step.

4. An implicit any-other entity exists when there is no any-other ACL entry. This implicit entry grants no permissions.

   Successful match: No permissions granted. End of evaluation process.

Evaluating unauthenticated requests

Tivoli Access Manager evaluates an unauthenticated user by granting the permissions from the ACLs unauthenticated entry.

The unauthenticated entry is a mask (a bitwise “and” operation) against the any-other entry when permissions are determined. A permission for unauthenticated is granted only if the permission also appears in the any-other entry.

Because unauthenticated depends on any-other, it makes little sense for an ACL to contain unauthenticated without any-other. If an ACL does contain unauthenticated without any-other, the default response is to grant no permissions to unauthenticated.

10.2.4 Protected object policies

A protected object policy (POP) specifies security policy that applies to an object regardless of what user or what operation is being performed. Each POP has a unique name and can be applied to multiple objects within a domain.

The purpose of a POP is to impose access conditions on an object based on the time of the access and to indicate whether the access request should be audited. Specifically, the conditions you can apply are:

- POP attributes, such as warning mode, audit level, and time-of-day.
- The authentication-strength POP allows for the configuration of step-up authentication to enforce stronger security for certain parts of the object space.
- The quality-of-protection POP implements privacy and integrity mechanisms such as encryption (SSL) and hash algorithms.
- The network-based authentication POP makes it possible to control access to objects based on the IP address of the client.

10.2.5 Authorization rules

Authorization rules are defined to specify conditions that must be met before access to a protected object is permitted. A rule is created using a number of boolean conditions that are based on data supplied to the authorization engine within the user credential, from the resource manager application, or from the encompassing business environment. The language of an authorization rule allows customers to work with complex, structured data by examining the values in that data and making informed access decisions. This information can be defined statically within the system or defined during the course of a business process. Rules can also be used to implement extensible, attribute-based authorization policy by using attributes within the business environment or attributes from trusted external sources.

A Tivoli Access Manager authorization rule is a policy type similar to an access control list (ACL) or a protected object policy (POP). The rule is stored as a text rule within a rule policy object and is attached to a protected object in the same way and with similar constraints as ACLs and POPs.

How authorization rules differ from ACLs and POPs

ACLs take a given predefined set of operations and control which users and groups have permission to perform those operations on a protected object. For example, a user's ability to read data associated with an object is either granted or denied by an ACL policy. POPs apply to all users and groups and control conditions that are specific to a particular protected object. For example, time-of-day access excludes all users and groups from accessing an object outside of the times set in the time-of-day policy.

Rules enable you to make decisions based on the attributes of a person or object and the context and environment surrounding the access decision. For example, you can use a rule to implement a time-of-day policy that depends on the user or

group. You also can use a rule to extend the access control capabilities that ACLs provide by implementing a more advanced policy, such as one based on quotas. While an ACL can grant a group permission to write to a resource, a rule can go a step further by enabling you to determine whether a group has exceeded a specific quota for a given week before permitting that group to write to a resource.

When to use authorization rules

In the Tivoli Access Manager authorization process, all three policy objects—the ACL, the POP, and the authorization rule—must permit access to a protected object before access to the object is granted. Authorization rules provide the flexibility needed to extend an ACL or POP by tailoring the security policy to your needs.

Although authorization rules can be used to extend the policy implemented by other Tivoli Access Manager policy types, they are not simply extensions of the existing policy types. An authorization rule is a policy type that is rich enough in functionality to replace the ACL and POP. However, using ACLs and POPs generally provides better performance. Therefore, use a rule to complement these policies instead of replacing them.

10.2.6 Authorization rules detail

The Access Manager authorization rules engine is implemented using an XSL parser. This, then, defines how the inputs to the rules engine must be specified.

The two inputs to an XSL parser are:

Document An XML document. In the case of the Access Manager authorization rules engine, this is a document, built internally, that contains all of the required ADI.

Stylesheet An XSL document. In the case of the Access Manager authorization rules engine, this is a document built from the configured rule for the object being accessed.

The output from an XSL parser is a new version of the document formatted using the stylesheet. In the case of the Access Manager authorization rules engine, the rules must be written in such a way that this *formatting* causes the output to be the access decision.

The diagram in Figure 10-5 on page 337 shows how the logical components of the rules engine are implemented using XML and XSL technology.

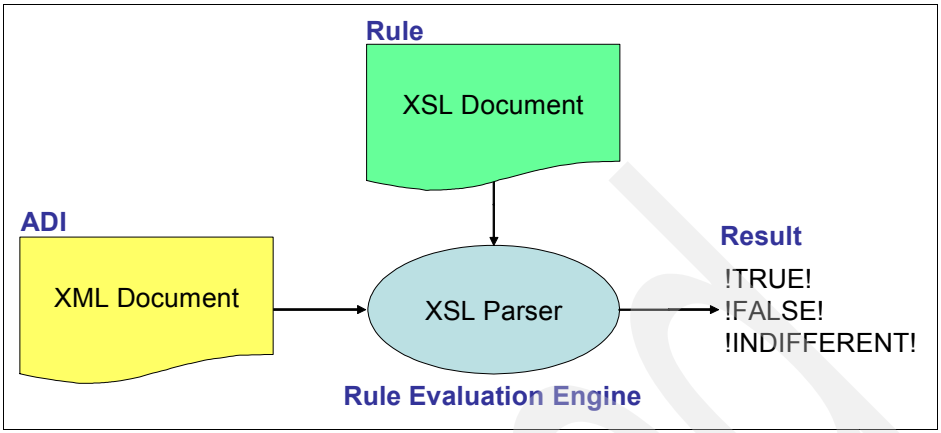


Figure 10-5 Authorization rules engine

The XSL parser *formats* the XML document containing the authorization decision information using an XSL formatted rule.

The rule must be written in such a way that the output is TRUE (permit access), FALSE (deny access), or INDIFFERENT (no opinion). Any other output is considered the same as FALSE (deny access).

10.2.7 External authorization capability

In some situations, the standard Tivoli Access Manager policy implementations of ACLs, POPs, and authorization rules might not be able to express all the conditions required by an organization's security policy. Tivoli Access Manager provides an optional external authorization capability to accommodate any additional authorization requirements.

The external authorization service allows you to impose additional authorization controls and conditions that are dictated by a separate, external, authorization service module.

Extending the authorization service

External authorization capability is automatically built into the Tivoli Access Manager authorization service. If you configure an external authorization service, the Tivoli Access Manager authorization service incorporates the access decision paths into its evaluation process.

Resource managers that use the authorization service, such as WebSEAL and any application using the authorization API, benefit from the additional, but seamless, contribution of a configured external authorization service. Any

addition to the security policy through the use of an external authorization service is transparent to these applications and requires no change to the applications.

The external authorization service architecture allows the full integration of an existing security service. An external authorization service preserves a company's initial investment in security mechanisms by allowing existing servers to be incorporated into the Tivoli Access Manager authorization decision-making process.

Imposing conditions on resource requests

An external authorization service can be used to impose more specific conditions or system-specific side effects on a successful or unsuccessful access attempt.

Examples of such conditions include:

- Causing an external auditing mechanism to record the successful or unsuccessful access attempt
- Actively monitoring the access attempt and causing an alert or alarm whenever unacceptable behavior is detected
- Conducting billing or micro-payment transactions
- Imposing access quotas on a protected resource

The authorization evaluation process

An authorization decision that incorporates an external authorization server takes place in the following manner:

1. If a trigger condition is met during the course of an access decision, the external authorization services that were configured for that condition are each called in turn to evaluate their own external authorization constraints. Invocation of the external authorization service occurs regardless of whether or not the necessary permission is granted to the user by the Tivoli Access Manager authorization service.
2. Each external authorization service returns a decision of permitted, denied, or indifferent. When indifferent is returned, the external authorization service has determined that its functionality is not required for the decision process and that it does not participate.
3. Each external authorization service decision is weighted according to the level of importance that its decision carries in the process. The weighting of individual external authorization services is configured when the service plug-in is loaded.
4. All authorization decision results are summed and combined with the decision made by the Tivoli Access Manager authorization service. The resulting decision is returned to the caller.

Figure 10-6 illustrates an authorization decision involving an application server and an external authorization service.

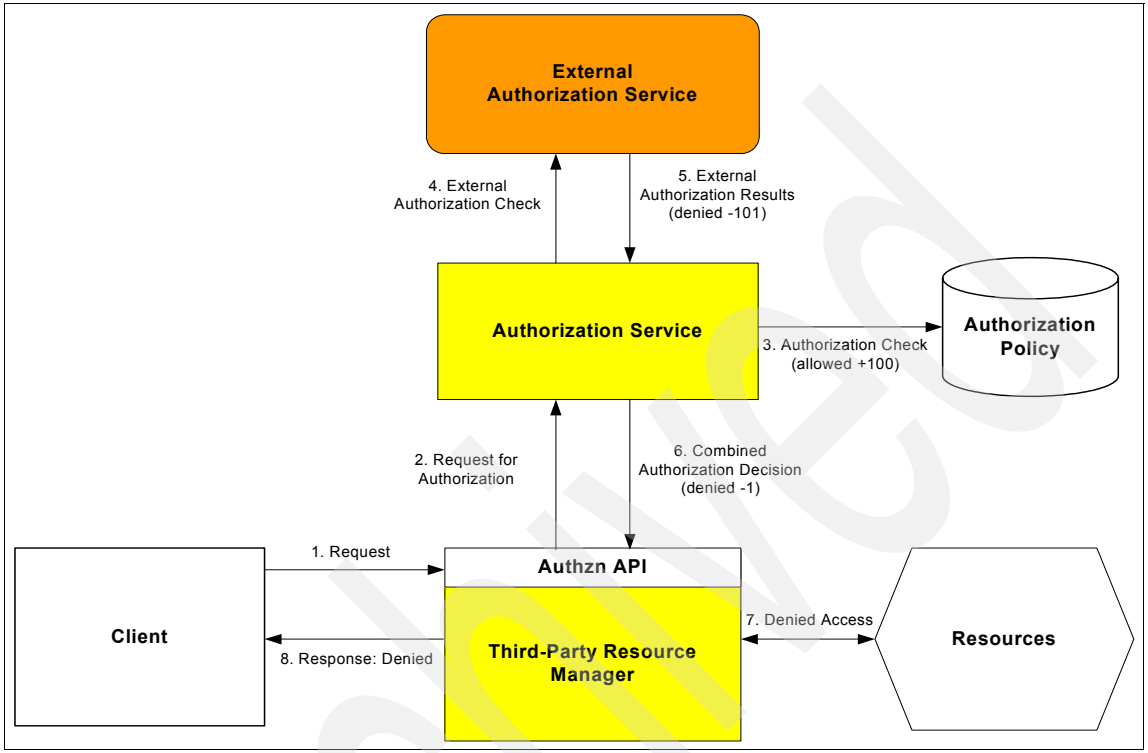


Figure 10-6 External authorization service with an application server

In this example, the purpose of the external authorization service is to impose a quota restriction on how often a photo-quality printer resource can be accessed.

The service implementation imposes a limit on the number of job submissions that any one person can make to this printer in one week. An external authorization service trigger condition was attached to the photo printer resource so that the external authorization service is invoked anytime that the photo printer is accessed.

The external authorization service was loaded with the default decision weighting of 101, which overrides any decision made by the Tivoli Access Manager authorization service, should it need to do so.

1. The resource manager server receives a request from a client for access to an online photo printing resource. The client is a member of the appropriate group GraphicArtists and so is normally permitted to submit jobs to the printer.

2. The application server first consults the Tivoli Access Manager authorization service to determine whether the requesting user has permission to submit jobs to the printer.
3. The authorization service checks the access permissions on the target requested object and compares these with the capabilities of the requesting user: `group GraphicArtists rx` In the ACL on the printer resource, the `x` permission grants any user in the GraphicArtists group access to the resource. Therefore, the authorization service grants the user permission to submit the job.
4. Because the photo printer resource is being accessed and an external authorization service trigger condition was attached to this object, a request is also made to the external authorization service configured for that trigger condition. The external authorization service receives all of the Access Decision Information (ADI) that was passed in with the original access decision check by the resource manager server.
5. The external authorization service consults a record of previous accesses made by this user. If the requesting user has not exceeded the quota for the week, it returns an access decision of “indifferent.” The implication is that the external authorization service is indifferent to the request and has no intention of participating in the access decision because its conditions for denying access have not been met. However, if the user has exceeded the quota, then the external authorization service returns a decision of “access denied”. For this example, it is assumed that the requester has exceeded the quota and that the external authorization service detects this and returns an “access denied” decision.
6. The Tivoli Access Manager authorization service receives the “access denied” result from the external authorization service. It then takes this decision and weights it with the default external authorization service weighting value of 101. The results of the external authorization service decision and the decision made by the Tivoli Access Manager authorization service are combined. The result is “access denied” because the result of the external authorization service (-101) outweighs that of the Tivoli Access Manager authorization service (100).
7. The resource manager server rejects the job submission to the photo printer resource.
8. The resource manager server returns a response to the caller to indicate that the job was rejected.

10.2.8 ADI

The authorization engine can gather Access Control Decision Information (ADI) from four sources for use when evaluating a rule:

- User credential entitlements
- Application context information passed in by the Tivoli Access Manager resource manager
- Tivoli Access Manager authorization engine context
- Dynamic ADI retrieval entitlement services

User credential entitlements

Additional entitlements data can be inserted as name and value attribute pairs (referred to as tag-value) into the client credential by a Tivoli Access Manager authorization client during the user authentication process or at any time during the process of the transaction. Tivoli Access Manager provides a credential attributes entitlement service that retrieves entitlements data from the user registry. Or, you can define your own entitlement services.

Any attribute added to the user credential can be used as ADI in a rule definition. There are also attributes that are built into the Tivoli Access Manager user credential when it is created by the authorization engine. Just as with attributes that can be added to the credential by the resource manager, the built-in credential attributes can be used in authorization rules. The built-in credential attributes include items of information, such as the user name (or the principal UUID) and the groups (or the group UUID) of which the user is a member.

Application context information

Authorization rules might require application context information to complete an evaluation. Context information includes information that is not an entitlement but is specific to the current transaction or operation. An example is a transaction amount, such as purchase price or transfer amount. This information is passed to the decision through the app_context attribute list of the azn_decision_access_allowed_ext() call. Tivoli Access Manager WebSEAL also uses this mechanism to pass the values of certain HTML tags and HTML request data (from a get or post request) into the access decision for use in a rule evaluation.

Authorization engine context information

Authorization engine context information is provided automatically by the authorization engine, if required, before the authorization rule is evaluated. The ADI provided by the authorization engine includes the name of the protected

object that is the target of the access decision and the string of operations that the requesting user wants to perform on the protected object.

The following attribute names are reserved for these data items:

- azn_engine_target_resource
- azn_engine_requested_actions

Dynamic ADI retrieval entitlement services

The final source for retrieving ADI is the dynamic ADI retrieval entitlements service. This class of azn entitlement services is designed to retrieve ADI from an external source. These services can be developed to retrieve ADI from an enterprise database containing employee, customer, partner, or inventory information. The dynamic ADI retrieval service is called to retrieve ADI at the time that the access decision is being made. Calling both at the same time has the benefit of being able to retrieve volatile data, such as quotas, at a time when its value is most current.

There are several methods available for retrieving Dynamic ADI:

- Resource managers
- Entitlement service
- Attribute Retrieval Service (ARS)
- Redirecting a user to ADI Provider site

Resource Managers

In order to provide *on demand* ADI to the rules engine, a resource manager must first register the information it is capable of supplying and then be able to respond to a request to supply it. Registration is done by specifying one or more prefixes that the authorization engine should use to identify *on demand* ADI that can be supplied by the resource manager.

If variables starting with these prefixes are found in authorization rules then a *deny* result is returned to the Resource Manager in response to the authorization request. However, an attribute is included with the result that specifies the ADI variables required. The resource manager must recognize that this attribute is present in the response from the authorization engine and then re-submit the access decision request with the required ADI. If the Resource Manager cannot supply the requested ADI (for whatever reason) then it must deny the user request.

Entitlement service

An entitlement service is a very generic plug-in that can be called by the Access Manager authorization service. Normally entitlement services are called as the result of an azn_entitlement_get_entitlements() call from a resource manager but, as in the case of ADI entitlement services, they can also be called by the authorization engine itself.

The input to an entitlement service is a user credential and an application context. The application context is simply an attribute list that can contain any information relevant to the entitlement service.

The output to an entitlement service is an attribute list. This is how the entitlement service passes back its results.

Attribute Retrieval Service

The final way to acquire ADI is really an extension of the dynamic ADI entitlement service. A dynamic ADI entitlement service is supplied out-of-the-box with Access Manager that can call the IBM Tivoli Access Manager Attribute Retrieval Service (ARS) to gather ADI.

The entitlements service formats the initialize, get_entitlements, and shutdown calls it receives from the authorization service into SOAP messages and sends them to a configured URL (which is the input interface for the Attribute Retrieval Service). It receives the response from the Attribute Retrieval Service and formats it back into an AM attribute list that it can return.

The IBM Tivoli Access Manager Attribute Retrieval Service, which is supplied with Access Manager as a J2EE application that runs in WebSphere Application Server, provides a framework for gathering ADI from external Information Providers or *Profiling Services*. A number of plug-ins for the Attribute Retrieval Service are provided for gathering information from certain providers. An interface is provided to enable additional custom plug-ins to be written that gather information from other providers. These are written in Java.

Redirecting a user to ADI provider site

In some cases, an ADI provider may need direct input from the user (or from the end users client, perhaps a client certificate or an SAML token) in order to be able to supply the requested ADI. To facilitate this, the dynamic ADI entitlement service has the capability to pass back a URL to Access Manager that is then passed back to the resource manager.

In response to receiving this URL (as an attribute to a *deny* response) the resource manager should direct the user to the URL so that they can go and interact directly with the ADI provider.

When the direct interaction with the ADI provider has been completed, they can (at some later time) return to the resource manager and repeat the original request. This time the request can be properly validated because the ADI provider is now in a position to provide the ADI required to evaluate the authorization rule.

10.3 Conclusion

Access Manager provides comprehensive and flexible methods of defining and enforcing security policy through resource managers. ACLs form the main component used to define policy. They define static relationships between users and groups, resources, and the actions a user can perform. POPs and extended attributes provide more flexibility, allowing for further criteria to be specified around access decisions.

Access Manager authorization rules give the ability to base access decisions on dynamic information about the current user. For example, access control lists do not provide the functionality to limit access to an object based on usage. To do this the current usage must be gathered in real time. Then make a decision by comparing it to some limit.

10.3.1 Guidelines for a secure protected object space

The following guidelines are suggestions for building and maintaining a secure protected object space:

- Set high-level security policy on container objects at the top of the object space. Set exceptions to this policy with explicit ACLs, POPs, and authorization rules on objects that are lower in the hierarchy.
- Arrange your protected object space so that most objects are protected by inherited, rather than explicit, ACLs, POPs, and authorization rules. Reduce the risk of an error that could compromise your network by simplifying the maintenance of your tree. Inherited ACLs, POPs, and authorization rules lower maintenance because they reduce the number of ACLs, POPs, and authorization rules you must maintain.
- Position new objects in the tree where they inherit the appropriate permissions. Arrange your object tree into a set of subtrees, where each subtree is governed by a specific access policy. You determine the access policy for an entire subtree by setting explicit ACLs, POPs, and authorization rules at the root of the subtree.

- Create a core set of ACLs, POPs, and authorization rules policies and reuse them wherever necessary. Because ACL, POP, and authorization rule policies are a single source definition, any modifications to the policy affects all objects associated with the ACL, POP, or authorization rule.
- Control user access through the use of groups. An ACL can consist of only group entries. Individual user entries are not required in the ACL when the users can be categorized into groups instead. Authorization rules can also be written to consider an individual's group memberships rather than the individual specifically. This can reduce the complexity of the rule logic considerably.

  Access to an object by individual users can be efficiently controlled by adding users to or removing users from these groups.

11

Application integration

An area of caution in IT security management is application-managed security. When different applications on different platforms driven by different project groups implement their own views of security functionalities, the result is an expensive, unmanageable turmoil that opens security holes instead of providing a strong access control solution. In developing new applications, we can start to build a solution that enables us to distinguish and differentiate between security and application functions.

Tivoli Access Manager aims to be a corporate access control solution. To reach this level of integration, there are a few application integration options that meet most application development platforms within today's e-business environments.

In this chapter we take a closer look at WebSphere, .NET, C and Java integration with Tivoli Access Manager.

11.1 Business requirements

Security is such a fundamental enabler of e-business that in the B2C and B2B markets, effective security can make the difference between owning the market or being just another competitor. The promise of e-business and its ability to create new revenue streams is predicated on the ability of these new business processes to reach these new markets and customers. These promises do not materialize if security issues are not addressed properly.

As enterprises extend their business applications to reach new markets and customers, security and trust issues take on paramount importance. This has always been true in core, mission-critical, intranet-based applications. This is even more true as these applications leverage the Internet's Web-based computing model for B2C and B2B.

As customers have moved to a Web-based computing model, some have found it very difficult to implement security on an application-by-application basis. With disparate applications that require disparate security approaches, it becomes clear very quickly that there is no security policy when there are numerous *islands of security* that cloud the picture. There is nothing nefarious about the islands of security approach; in fact, it can be a natural evolution for customers, because many products come with some form of security built in. But when the islands begin to diminish, the ability to clearly manage security according to policy for your organization decreases, so there is tremendous value in securing applications in a way that is consistent and compatible with securing applications and application-components running on other middleware and platforms in the enterprise.

For this scenario, we define the following business requirements for existing as well as new e-business applications based on WebSphere family products:

- Reduced costs of implementing and maintaining proprietary security solutions (islands of security)
- Fast time-to-production
- Reduced cost and complexity of application development
- Consistently managed end-to-end security (from browser to Web application) in order to mitigate risks of fraud
- Applications developed according to standards and standard architectures in order to achieve independence of specific vendor solutions

11.2 Security design objectives

Based on business requirements, we define the following security design objectives to be achieved by integrated solutions:

- Simplification of application development and off-loading the security policy of the application
- Simplification of system administration by maintaining a consistent security model across applications and related systems

Regarding the implementation of an access control subsystem, the systems fall into one of the following three categories.

Category 1 systems implement and enforce their own authorization decision processes based on security policies defined in proprietary formats, as shown in Figure 11-1.

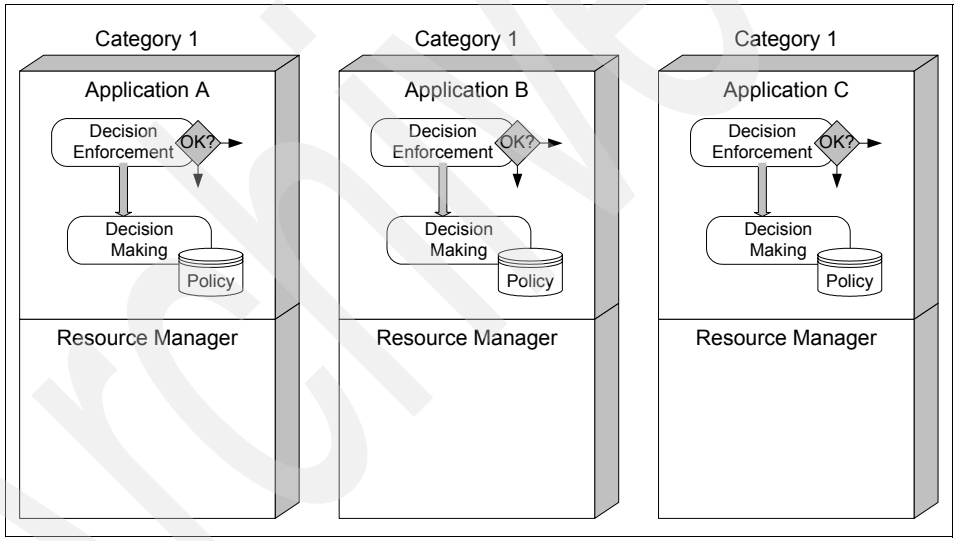


Figure 11-1 Category 1 systems

Obviously, these are home-grown applications that may even precisely reflect the existing security policies. However, the risk is rather high that such implementations go outside the designed limits in rapidly extending and changing IT environments. As security decision making, as well as their enforcement, is implemented inside the application, any change in the policies must be reflected in the application code. Moreover, it becomes difficult to ensure that all category 1 systems enforce the same policy. As an outcome in category 1 systems, maintenance costs for updating security in the applications are rising, and valuable development time is spent writing security, not business functions.

Category 2 systems address this issue by offloading the authorization decision-making process from the application to a resource manager, as shown in Figure 11-2. The resource manager takes over the role of providing the requested resources to the application and decision making process. If a resource is requested by the application, it calls the authorization decision-making process residing in the resource manager. The resource manager consults its policy database and provides the application with a simple yes-or-no decision. It is then up to the application to enforce the received decision and provide information as the user requests, or decline it. A series of subsequent authorization decision calls may be necessary to come to the final go-or-no-go decision.

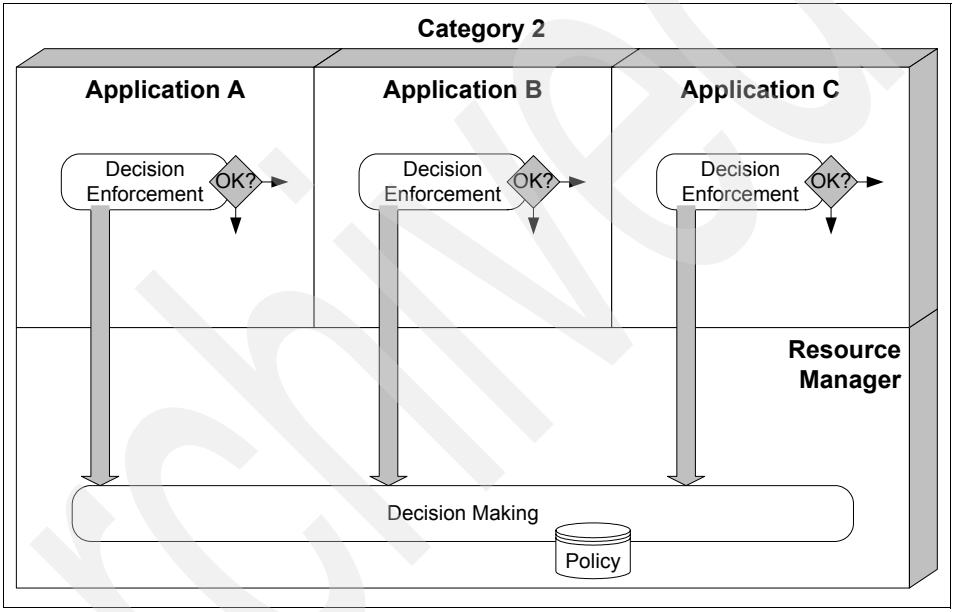


Figure 11-2 Category 2 systems

By separating the two functions (decision making and decision enforcement), it is much easier to achieve reusability of the decision making processes and consistency of the policies. The Tivoli Access Manager software architecture supports this category, providing a uniform framework for authorization decisions and Open Group's Authorization application programming interface (API) for its querying. Decision enforcement takes place on the blades in order to meet the needs of distributed applications acting in disparate environments with different security requirements.

However, an application based on a system of category 2 has to implement its own decision enforcement. If it is not standardized (for example, based on

Authorization API provided by Open Group), it also must implement the decision requester, which may be considered to be more error-prone.

To avoid this problem, systems of category 3 rely on mechanisms provided by a resource manager and have no need to even maintain decision enforcement, as shown in Figure 11-3.

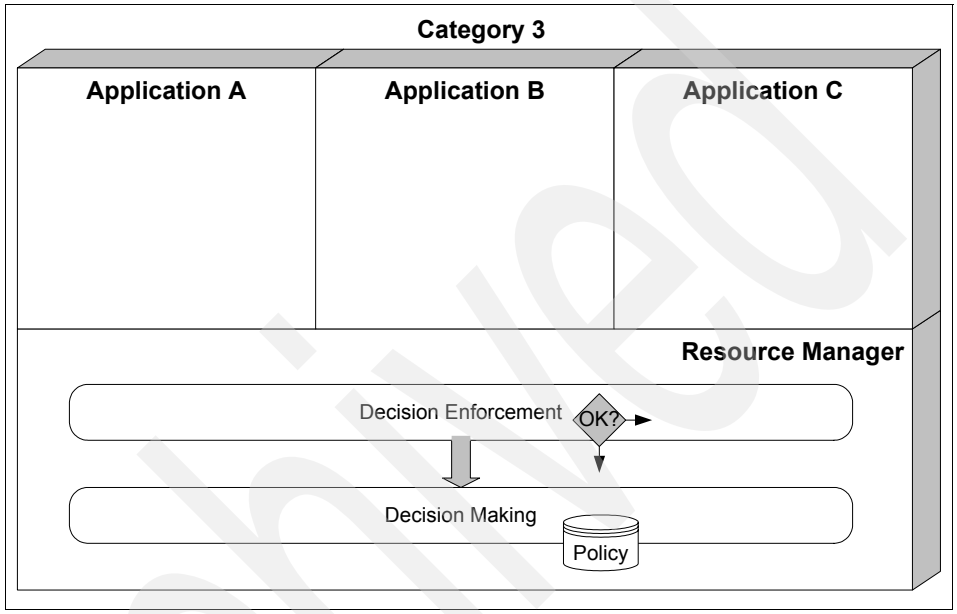


Figure 11-3 Category 3 systems

An application that falls in this category is a Web server providing access to files in a defined directory. In a simple case, it uses security mechanisms of the operating system that act as a resource manager. If a user is requesting an HTML document, the operating system's file permissions are decisive for granting access. The application (Web server) requests a resource (file), managed by the operating system. While serving the request, the operating system makes a decision based on the permission attributes (policy) of the requested file and, if allowed, provides access to the file (decision enforcement) by the Web server. Applications based on Enterprise Java Beans (EJB) work in a similar way.

This approach works when applications reside together with the resource manager on the same system. It becomes much more difficult to manage if multiple applications of the same kind are distributed through the IT environment and communicate with the same resource manager. Moreover, as soon as a need arises to establish security policies throughout applications based on

different resource managers of different kinds, a new consolidation layer is required.

As shown in Figure 11-4, Access Manager provides that uniform authorization framework, which enables you to consolidate the decision-making process based on a consistent policy database.

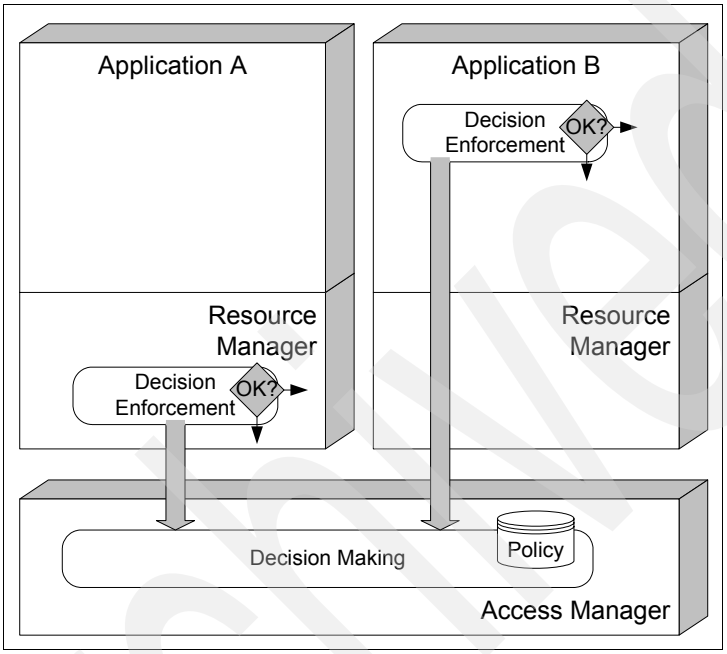


Figure 11-4 Policy enforcement based on consistent decision making

11.3 WebSphere Application Server security

WebSphere Application Server Version 6.0 is a Java 2 Enterprise Edition compliant Java application server.

Here we concentrate on the J2EE 1.4 security features implemented in WebSphere V6.0:

- Role-based security
- Declarative security
- Programmatic security

Role-based security

One of the goals of the J2EE specification was to lessen the burden of application security on application developers. Previously, if a portion of code could only be executed by particular types of users, the code itself had to handle the authorization, often right within the business logic. For example, if only managers were allowed to execute a function, then each user attempting to call that function would have to be identifiable as a manager. This might require a lookup in an employee database to determine the user's employee type or group type. This led to the development of category 1 systems, as described in Figure 11-1 on page 349.

J2EE attempt to move this security burden to the application *assemblers* and *deployers*. It enables them to define security roles, sets of permissions for access to Web resources, and specific EJB methods. The use of roles provides a level of indirection that enables the subsequent assignment of those roles to users and groups to be done at application installation time, rather than during development. It also allows security constraints within modules developed by different teams to be resolved at assembly, deployment, or installation time.

The J2EE specification defines a security role as a logical grouping of users that is defined by an Application Component Provider or assembler. It is then mapped by a deployer to security identities (for example, principals or groups) in the operational environment. A security role can be used either with declarative security or with programmatic security. Thus, WebSphere's security is role-based. This authorization model is shown in Figure 11-5.

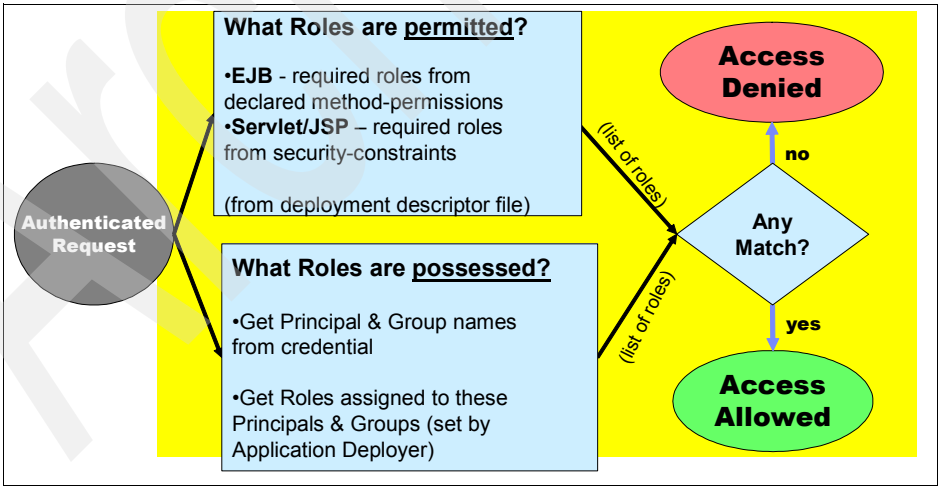


Figure 11-5 Role-based authorization model

Declarative security

The declarative security mechanisms, as part of J2EE, are stored in a document called deployment descriptor using a declarative syntax. Global security roles for a WebSphere application are stored in the XML deployment descriptor. Security roles for WebSphere components are stored in their corresponding deployment descriptors inside the EAR, Java archives (JARs), and Web archives (WARs).

WebSphere uses method permissions, introduced in the EJB 1.1 specification, to describe security roles for EJBs. For a particular EJB resource, method permissions are the association of role names with the sets of methods, based on what types of permissions should be required to invoke the methods.

Example 11-1 demonstrates a slightly abbreviated sample role description for EJB methods within an ejb-jar.xml deployment descriptor. Only a user who can be mapped to the security role Teller is allowed access to the methods getBalance and getLastTransaction of the bean AccountBean.

Example 11-1 Method permissions in the ejb-jar.xml deployment descriptor

```
<method-permission>
       <role-name>Teller</role-name>
       <method>
          <ejb-name>AccountBean</ejb-name>
          <method-name>getBalance</method-name>
       </method>
       <method>
          <ejb-name>AccountBean</ejb-name>
       <method-name>getLastTransactions</method-name>
       </method>
</method-permission>
```

If WebSphere security is enabled and EJBs have no method at all configured with security, then the default is to grant access to the EJB methods. If WebSphere security is enabled and at least one method has a security constraint, then the request to the EJBs is denied. This kind of behavior is different compared to the Web modules' components. By default, access is allowed to all Web resources. Parts of the Web resources can be protected using security constraints.

For a particular Web resource (servlet, JSP, and URL), security constraints are the association of role names with the sets of HTTP methods, based on the types of permissions that should be required to access the resource. These are defined in the WAR's deployment descriptor. Example 11-2 on page 355 shows a WAR deployment descriptor that restricts access to any URL containing the URL-pattern /sales/ to the methods HTTP-POST and HTTP-GET and to users, that can be mapped during runtime to a security role called SalesPerson.

Example 11-2 Security constraints and permissions in a WAR deployment descriptor

```
<web-app>
       <display-name>Retail Application</display-name>
       <security-constraint>
          <web-resource-collection>
             <web-resource-name>SalesInfo</web-resource-name>
             <url-pattern>/sales/*</url-pattern>
             <http-method>POST</http-method>
             <http-method>GET</http-method>
<          /web-resource-collection>
          <auth-constraint>
             <role-name>SalesPerson</role-name>
          </auth-constraint>
       </security-constraint>
</web-app>
```

Figure 11-6 depicts declarative security based on security roles. The objects (EJB methods, static Web pages, servlets, and JSPs) are protected by method permissions or security constraints. Permissions and constraints are mapped to security roles. The deployer grants access to roles for users and groups.

So far, there is no need for a developer to implement a single line of code to achieve security.

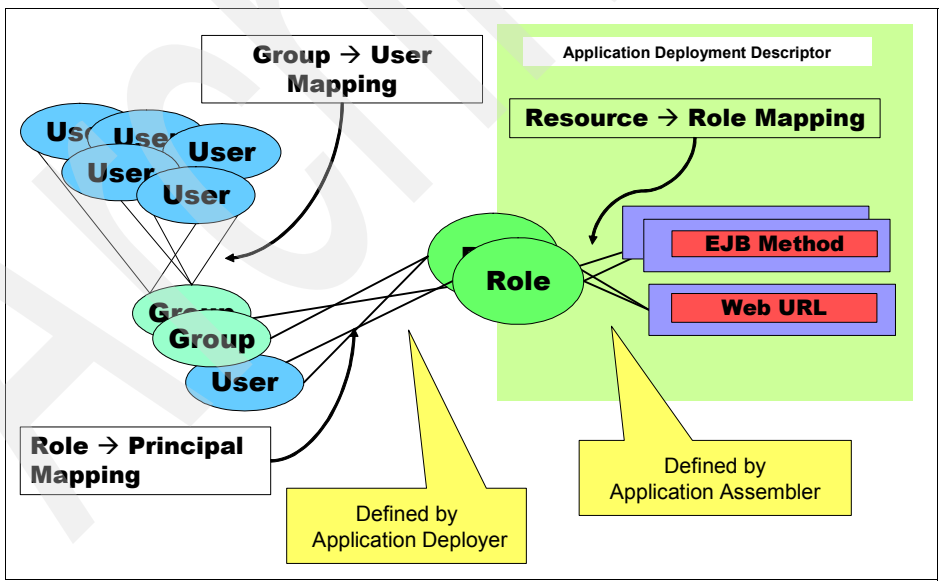


Figure 11-6 Role-based security

Programmatic security

Declarative security is not always sufficient to express the security model of the application. Using a payment transaction example: A customer has to have access to a bean method in order to transfer money. If he is granted access, he can perform any transaction he wants. In order to limit the amount of money that can be transferred by this user, the application must have knowledge about the role of the customer.

Developers can check security constraints programmatically using the name of the role. The API for programmatic security in J2EE consists of two methods of the EJB EJBContext interface and two methods of the servlet HttpServletRequest interface:

- isCallerInRole (EJBContext)
- getCallerPrincipal (EJBContext)
- isUserInRole (HttpServletRequest)
- getUserPrincipal (HttpServletRequest)

These methods enable components to make business logic decisions based on the security role of the caller or remote user. In our example, the application may use the method isUserInRole to verify whether the user is allowed to transfer the amount of a given sum. Another possibility would be to use the method getUserPrincipal to use the user's principal name as a key to get more authorization information stored elsewhere.

To summarize, WebSphere Application Server authorization uses a role-based model. WebSphere Application Server treats a role as a set of permissions to access particular resources.

11.3.1 Java Authorization Contract for Containers

Java Authorization Contract for Containers (JACC) was introduced in J2EE 1.4 specification to address some problems and limitation of earlier definitions:

- All access decisions made by the application server, unless proprietary interfaces 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, 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 is a named collection of these permissions.

Please note that JACC does not specify a standard interface for principals 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 11-7 shows these relationships.

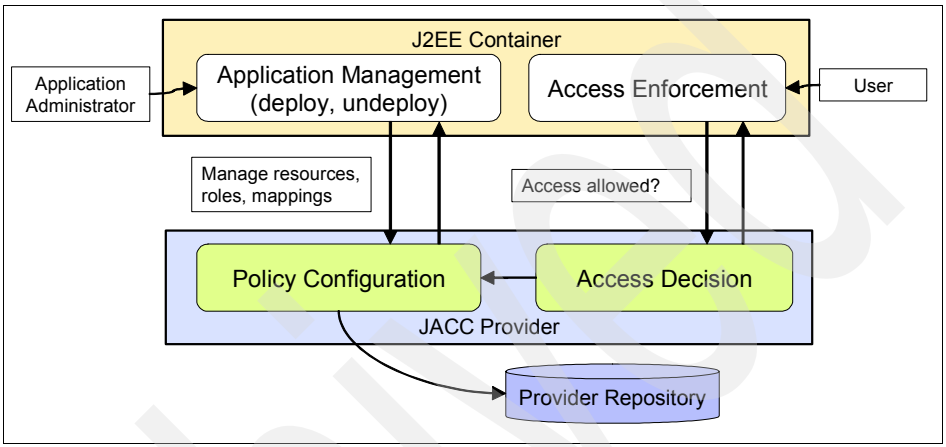


Figure 11-7 Java Authorization Contract for Containers

11.4 Access Manager and WebSphere integration

Providing a standard-based authorization framework for WebSphere applications, Tivoli Access Manager supports the Java 2 security model as well as the Java Authentication and Authorization Services (JAAS) and Java 2 Enterprise Extensions (J2EE).

Integrating WebSphere and Access Manager adds WebSphere resources to the significant list of elements that can be managed via Tivoli Access Manager's consistent authorization policy, and it also adds to WebSphere applications the benefits that accrue in an Access Manager protected environment. The examples of this discussed in the previous chapters include URI-based access control, availability and scalability characteristics inherent in Access Manager implementations, and the ability to support many authentication mechanisms without any impact to the target application and Web single sign-on, which are fully applicable for WebSphere Application Server.

The integration of WebSphere Application Server and Access Manager offers the following additional options/possibilities:

- Shared user registry
- Web single sign-on using:
  - Tivoli Global Sign-On (GSO) junctions
  - Trust Association Interceptor Plus (TAI++)
- Application integration utilizing:
  - Authorization Application Programming Interface (aznAPI)
  - JAAS
  - PDPermission
  - J2EE security

11.4.1 Shared user registry

Both WebSphere and Access Manager need a user registry to store user information, such as IDs and passwords. The first area of integration is for both products to use the same user registry, and so have a single, common set of users defined to both WebSphere and Access Manager. They each support a number of Lightweight Directory Access Protocol (LDAP) servers for this purpose. Obviously, to share the same user registry you must choose a server that both products support. The most common is to use Tivoli Directory Server.

Administration considerations

WebSphere has no interface for administering users in an LDAP server, so you have to use the tools that are provided with the LDAP server product. Access Manager, on the other hand, does have tools: the command line interface and the Web Portal Manager Administrator Console.

When sharing the same directory, WebSphere Application Server has to be configured to meet Tivoli Access Manager LDAP requirements. The changes to be aware of are:

- Anonymous access to LDAP is no longer permitted. WebSphere must be configured with a Bind Distinguished Name.
- The default WebSphere group filter defined for the particular LDAP server must be updated.
- LDAP access control lists (ACLs) are modified. You require a special privilege to be able to perform a directory search. WebSphere bind distinguished name must be able to perform directory searches to retrieve users and groups and populate user and group-selection lists.

11.4.2 Single sign-on

Single sign-on between Access Manager and WebSphere Application Server 6 can be achieved using two different mechanisms:

- GSO junctions
- Trust Association Interceptor Plus

Please note that WebSEAL still supports LTPA integration, but it's only supported for WebSphere 5.x applications running on WebSphere Application Server 6. Trust Association Interceptor Plus is the preferred method, as it supports both previous and current application versions under WebSphere Application Server 6 and also WebSphere Application Server 5.11.

GSO junctions

Access Manager's Global-Sign-On provides a mapping between the primary user identity (used for login to WebSEAL) and another user ID/password that exists in another user registry.

In a pure WebSphere environment, accessing a protected URL causes an HTTP 401 challenge to the browser. The user enters authentication details (user ID and password), and this information is passed in a basic authentication (BA) header back to WebSphere. WebSphere Application Server then uses the authentication information to perform an LDAP-bind to authenticate the user.

The different GSO options and capabilities are described in detail in 9.5.1, "Tivoli Global Single Sign-On lockbox" on page 307.

Trust Association Interceptor Plus (TAI++)

In a customer's corporate distributed environment, the Access Manager security architecture utilizes a reverse proxy security server, WebSEAL, as an entry point to all service requests. The intent of this implementation is to have WebSEAL as the only exposed entry point. As such, it authenticates all requests that come in and provides course-granularity junction point authorization.

When WebSphere is used as a back-end server it further exploits its fine-grained access control. WebSEAL can pass to WebSphere an HTTP request that includes the credential of the authenticated user. WebSphere can then use these credentials to authorize the request.

The *Trust Association Interceptor Plus* provides a WebSphere interface with third-party objects. It intercepts requests issued by trusted proxy servers, such as WebSEAL. These objects are collectively known as *Trust Association Interceptors* or simply *interceptors*.

TAI++ implies that the WebSphere security application recognizes and processes HTTP requests received from WebSEAL. WebSphere and WebSEAL engage in a contract in which the former gives its full trust to the latter, which means that WebSEAL applies its authentication policies on every Web request that is dispatched to WebSphere.

This trust is validated by the interceptors that reside in the WebSphere environment for every request received.

Using TAI++

When using *Trust Association Interceptor Plus*, WebSEAL authenticates the user, acquires credentials for the user from the user registry and possibly authorizes the request at URL level. With a successful authorization, WebSEAL augments the request with an additional HTTP header (iv-creds) that contains the user's credentials. It also changes the password contained in the Basic Authentication header so it matches a configured SSO user.

This request is sent to WebSphere Application Server, who calls a TAI method to determine whether the request is from a perimeter authentication service that has already authenticated the user, to establish trust with the perimeter authentication server and retrieve the credentials. This method establishes trust with WebSEAL by checking if the Basic Authentication header contains the correct password for the configured SSO user. It is done by calling an Access Manager Authorization Server to make this decision.

The *iv-creds* header is then extracted from the request and used to construct a PDPrincipal object. A credential object containing user and group information is constructed from information contained in the PDPrincipal. The Principal and the Credential objects are inserted into a JAAS Subject which is returned from the call. At this point WebSphere Application Server has valid credentials that it can use for making authorization decisions in the usual J2EE manner. In addition, the Subject now contains the PDPrincipal object which application code can access if needed.

If a remote call is made to an EJB on a downstream server the credential information (that was initially extracted in the TAI) is serialized and sent to the downstream server. In addition, if a cluster is in place, the serialized Subject is also replicated horizontally using the WebSphere Application Server propagation framework.

Important points to note are:

- WebSEAL needs to insert the iv-creds header into the request, not the iv-user header.
- TAI++ does not directly contact LDAP unlike the previous TAI version. It instead contacts the Access Manager Authorization Server which validates

the SSO password to establish trust with WebSEAL. This means that additional configuration is required on the WebSphere Application Server side to ensure that the TAI can reach the Access Manager Authorization Server.

- The Credential object inserted into the Subject by the TAI means WebSphere Application Server does not have to perform any additional user registry searches as part of the authentication process.
- Native WebSphere Application Server Authorization or Access Manager JACC authorization will work with this Subject.

The TAI++ overall process is shown in Figure 11-8.

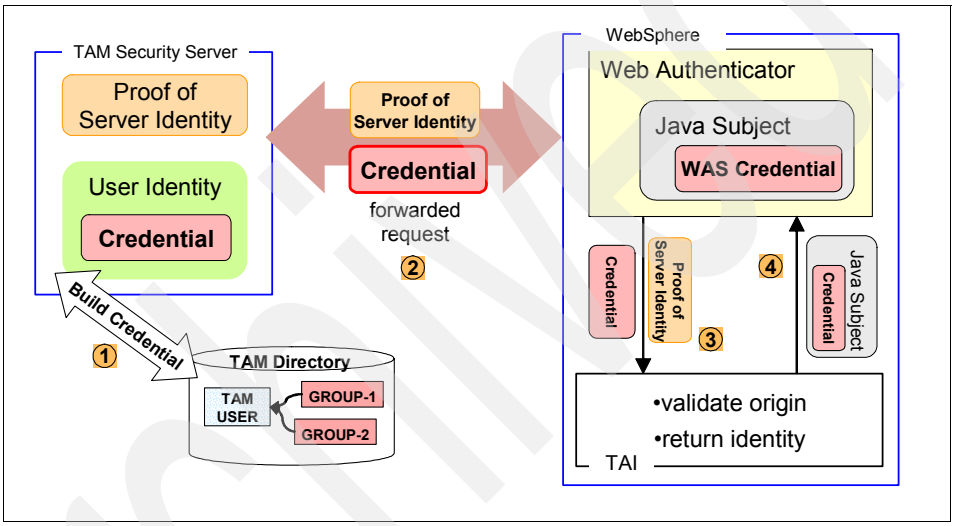


Figure 11-8 TAI++ overall process

Trust Association Interceptor Plus is transport agnostic and processes HTTP and HTTPS requests in an identical manner. Please be aware that trust between WebSEAL and WebSphere Application Server cannot be established using mutually authenticated SSL sessions and can only be established by verifying the SSO password. No checking of certificates is performed by the TAI.

The TAI++ logical architecture is shown in Figure 11-9.

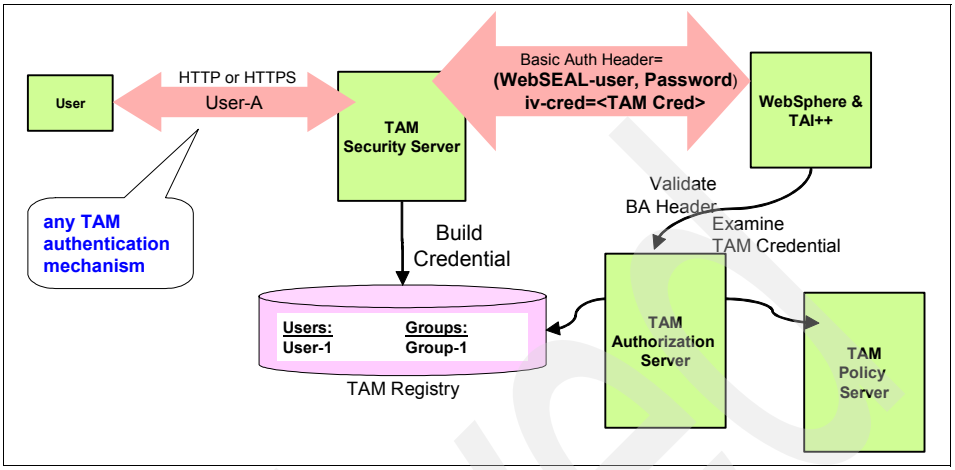


Figure 11-9 TAI++ logical architecture

Summary of how TAI++ works

To sum up how Trust Association works using the WebSphere Administration Applications:

1. The browser requests a URI that WebSEAL recognizes to be a protected resource.
2. WebSEAL prompts the user to provide a user ID and password (this can be either via Basic Authentication challenge or via a Custom Form).
3. WebSEAL authenticates the user.
4. After proper authentication and coarse-grained authorization, WebSEAL forwards a modified HTTP request to the back-end WebSphere server.
5. TAI++ then extracts the value of the iv-cred and basic authentication http header, validates it against Access Manager Authorization Server, and returns this as the authenticated user that should be used by WebSphere authorization.

How to select the SSO option

In fact, if you assume that Access Manager and WebSphere share a user registry, then GSO would be the last choice for SSO. Instead, using the Trust Association Interceptor Plus (TAI++) would be the preferred solution. GSO is only an option when WebSEAL and WebSphere rely on different user registries. In

this case you may need to supply a different user ID and password combination for the user to WebSphere that is meaningful to the WebSphere user registry.

Otherwise, we recommend the TAI++ option, because it is easy to configure and maintain. There is no key distribution or periodic update required. TAI++ is also the method used when WebSphere supports integration with third-party reverse proxy security servers in general.

11.4.3 User mapping for WebSphere J2EE Connector Architecture

The J2EE Connector Architecture, sometimes called J2CA, JCA, J2C, or JCX, defines a standard architecture for connecting the Java 2 Platform, Enterprise Edition (J2EE) to heterogeneous enterprise information systems (EIS). Examples of EIS include Enterprise Resource Planning (ERP), mainframe transaction processing (TP), and database systems.

The connector architecture enables an EIS vendor to provide a standard resource adapter for its EIS. A resource adapter is a system-level software driver that is used by a Java application to connect to an EIS. The resource adapter plugs into an application server and provides connectivity between the EIS, the application server, and the enterprise application.

Accessing information in EIS typically requires access control to prevent unauthorized accesses. J2EE applications must authenticate to the EIS to open a connection to it. Figure 11-10 depicts the J2CA architecture.

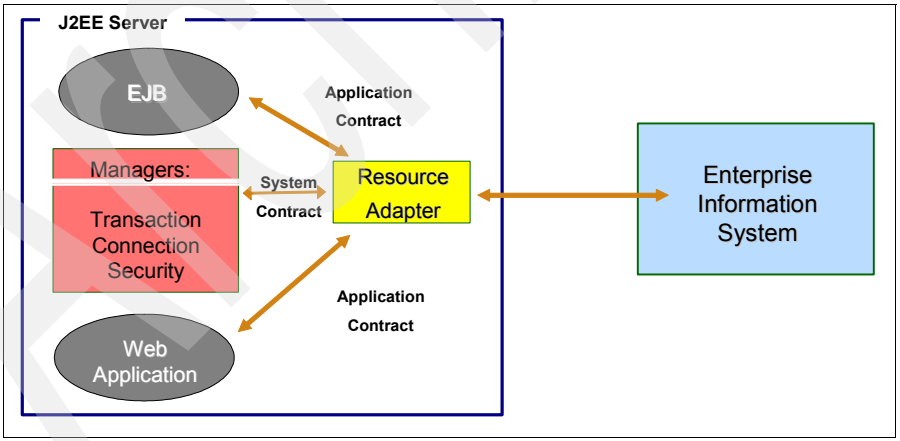


Figure 11-10 J2EE Connector Architecture - J2CA

As seen in Figure 11-11, the user identity flows from the point of authentication to the back-end EIS. The user identity can be propagated or mapped for each transaction, without requiring the application logic for this mapping when in *Container managed mode*.

The J2CA specifies the use of a JAAS login module for:

- Principal mapping
- Creation of principal's credentials

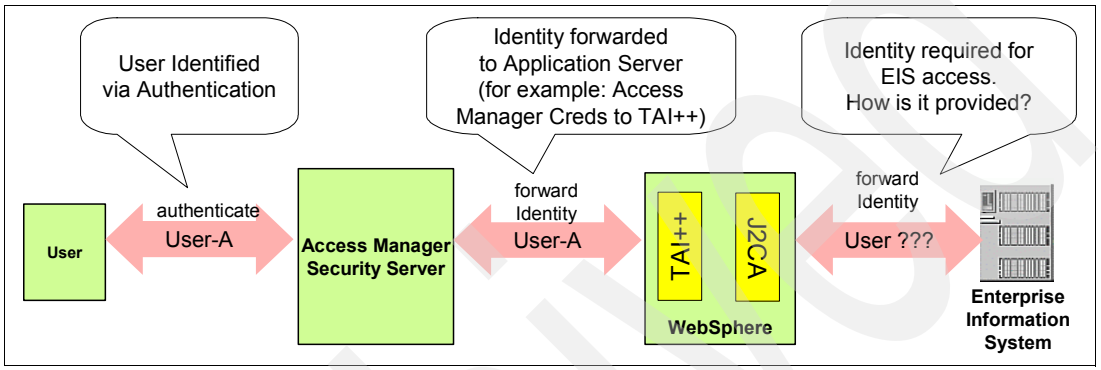


Figure 11-11 J2CA security objective: Provide identity to enterprise information system

The EIS sign-on can be managed by the application or the container. When application management is in use, the application provides the identity and authentication data. When container management is in use, the JAAS login module provides the identity and authentication data.

Tivoli Access Manager provides an *Access Manager Global Sign-on Principal Mapper,* which is configured as a J2CA JAAS login module. It locates an Access Manager GSO entry using the Access Manager user and the GSO target. Access Manager can use the currently authenticated user (subject in the security context), or part of the authDataAlias string, that is a property of the ConnectionFactory.

The Access Manager user and GSO target uniquely identify the GSO entry. The GSO entry contains the user ID and password to be provided by the resource adapter to the Enterprise Information System.

Figure 11-12 shows the Access Manager GSO principal mapping for J2CA.

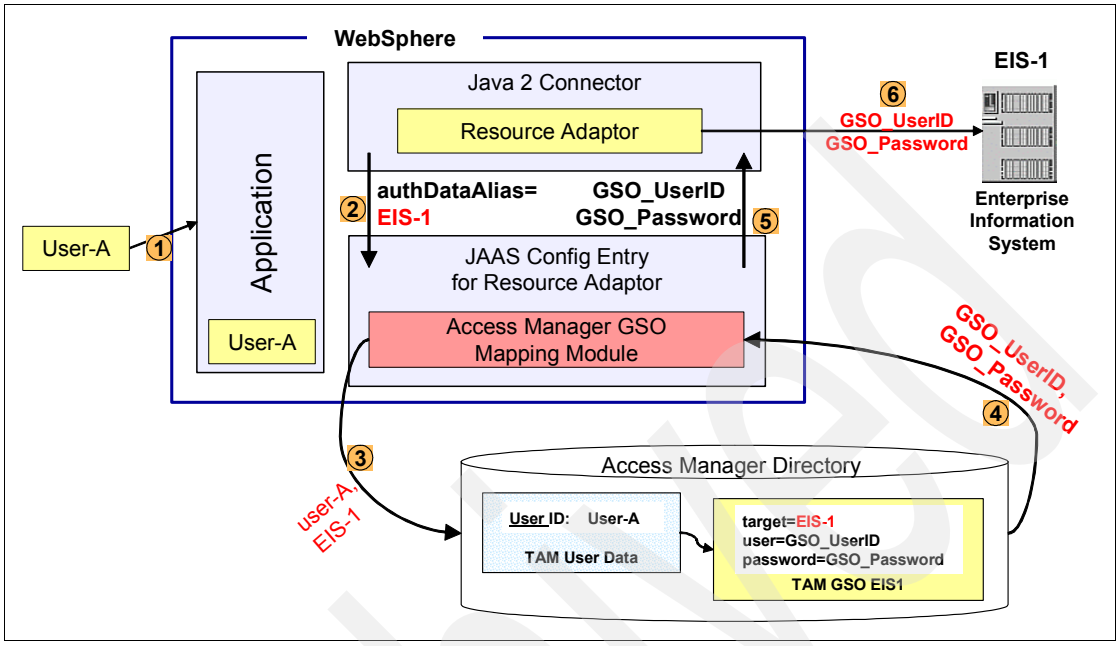


Figure 11-12 Access Manager GSO principal mapping for Java 2 connectors

Access Manager JACC Provider

The Access Manager JACC Provider is shipped with WebSphere Application Server 6.0. It replaces the previous Access Manager for WebSphere (AMWAS) component that was shipped with Access Manager.

When using the Access Manager JACC provider and a user attempts to access a protected resource (one that requires some role), the access enforcement point for the container authenticates the user (if necessary) to create a WebSphere credential. After this JACC interfaces are called passing the resource, user, and type of access required. The Access Manager JACC provider then uses a local copy of the Access Manager Policy Database to determine if the user possesses a role that grants access.

All Application Management events are passed to the Access Manager JACC provider (via the WebSphere console). The Access Manager JACC Provider

communicates with the Access Manager Policy Server to persist the J2EE security configuration data:

- Resource to Role mappings – for example, the URLs and EJB methods that have security requirements and the roles that are allowed access.
- Role to Principal mappings – for example, which users/groups are granted the roles required by the application's resources.

This data is first created in the Policy Server's master policy database and then replicated to every WebSphere Application server that is configured into the same Access Manager secure domain. All WebSphere servers in that cell should be configured to use the same Access Manager domain by using the Access Manager JACC configuration built into the WebSphere Application Server Admin Console.

Figure 11-13 shows both Application Management and Access Enforcement between WebSphere Application Server and Tivoli Access Manager using Access Manager JACC Provider.

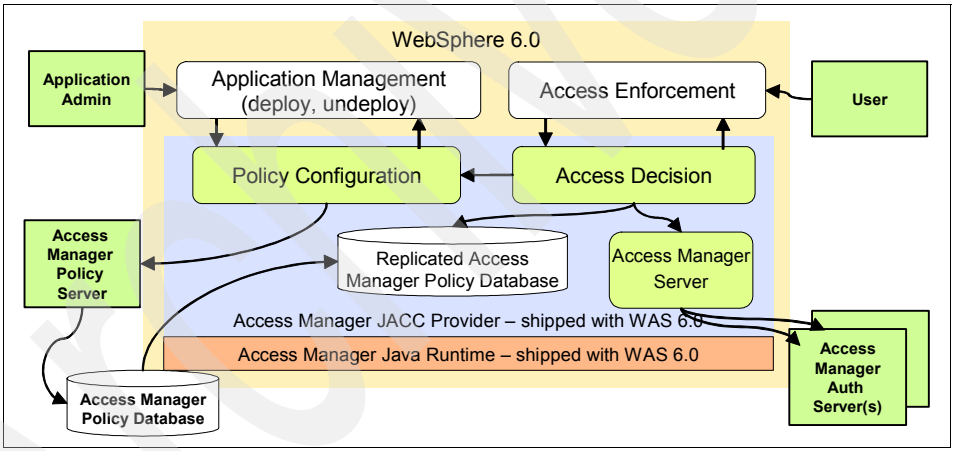


Figure 11-13 Tivoli Access Manager JACC Provider

The benefits of the Access Manager JACC Provider over Access Manager for WebSphere Application Server are as follows:

- Easier deployment
  - No additional Access Manager installation required
  - Access Manager JACC Provider only needs configuration
    - WebSphere Admin Console or wsadmin CLI

- Integrated policy configuration
  - Access Manager for WebSphere Application Server migration utility eliminated
  - Application deploy/undeploy exports policy to Access Manager
- Dynamic Policy Updates
  - Role-to-Principal changes do not require application restart. New policy is effective as soon as Access Manager Policy Database is replicated
- Integrated Administration
  - Role-to-Principal mapping available on WebSphere Admin Console

Please note that existing applications must be re-deployed or migrated to benefit from Access Manager JACC provider.

11.5 Access Manager and .NET Integration

IBM Tivoli Access Manager for .NET provides a standard-based authorization framework for .NET. Integrating .NET framework and Access Manager adds .NET resources to the significant list of elements that can be managed via Tivoli Access Manager's consistent authorization policy, and it also adds to .NET applications the benefits that accrue in an Access Manager protected environment. The examples of this discussed in the previous chapters include URI-based access control, availability and scalability characteristics inherent in Access Manager implementations, the ability to support many authentication mechanisms without any impact to the target application, and Web single sign-on, which are fully applicable for .NET environment.

The integration of .NET and Access Manager offers the following additional options/possibilities:

- SSO from Access Manager Web security servers to ASP.NET applications
  - Accepts Access Manager user ID or credential
  - Authenticates traffic origin
- Role membership evaluation using Access Manager policy
  - Declarative role security
    - Application configuration enforced by application server
  - Programmatic role security
    - Application API call – "does user possess this role?"

- Web service security
  - Client-side authorization and Identity propagation (via HTTP headers)
  - Server-side authentication and authorization
    - HTTP header or SOAP WS-Security header (UsernameToken)
- Exposure of Access Manager APIs to .NET applications
  - Access Manager Authorization & Administration APIs
  - API-level help for MS Visual Studio® .NET

Within the .NET framework, the common language runtime (CLR) is responsible for run-time services such as:

- Language integration
- Common language runtime
- Security enforcement
- Memory process and thread management.

Access Manager for .NET exposes Access Manager APIs to the .NET common language runtime level, thereby making the functions available to all .NET languages such as:

- Managed C++
- C#
- Visual Basic .NET

11.5.1 Single sign-on

An ASP.NET application can be configured to use the Access Manager Authentication Module to achieve SSO from Access Manager Web security servers. The capabilities and options are the functional equivalent of the Trust Association Interceptor (TAI) for WebSphere.

The input request carries the user's identity in an HTTP header. The value can be a simple user ID string or the Access Manager credential created by the Web security server (these are typically carried by http_iv_user and http_iv_creds respectively).

A user ID that represents the Web security server itself can be configured to provide a trust basis for the request. If configured, the password field of the basic authentication header must be the password of this "Web security server" user.

In addition to the trust-basis provided by the basic authentication header password, the Access Manager Authentication Module can also confirm that an

SSL session was used to transport the request and that a client-side certificate was used for the SSL connection. This provides an additional level of trust that the user identity came from a trusted source.

The Access Manager Authentication Module uses the input identity to create an *AccessManagerPrincipal* object that is an implementation of the .NET *IPrincipal* interface. This identity is placed in the *HttpContext* where it is used by the .NET server for declarative role-based access decisions and is available to the application for programmatic role-based decisions.

Figure 11-14 shows Web single sign-on to the ASP.NET environment. Note that http headers never go to the browser. They only exist between the Access Manager Security Server and the IIS Web server.

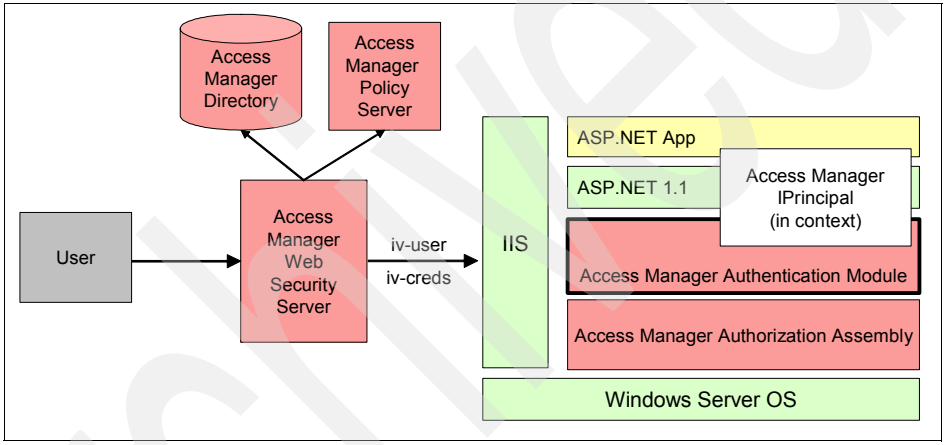


Figure 11-14 Access Manager Web single sign-on to ASP.NET environment

11.5.2 Role-based authorization in .NET

.NET uses role-based authorization. Access Manager for .NET provides *Principal to Role mapping* for .NET. With Access Manager for .NET in use, authorizations in .NET applications benefit from the central authorization framework that Tivoli Access Manager provides.

Authorization from .NET applications

Access Manager for .NET exposes the Access Manager aznAPI to the .NET environment, providing a standard-based authorization framework for .NET applications. This is done by an Access Manager assembly deployed in the .NET environment. Figure 11-15 on page 370 shows how this assembly integrates a .NET application with Tivoli Access Manager authorization framework.

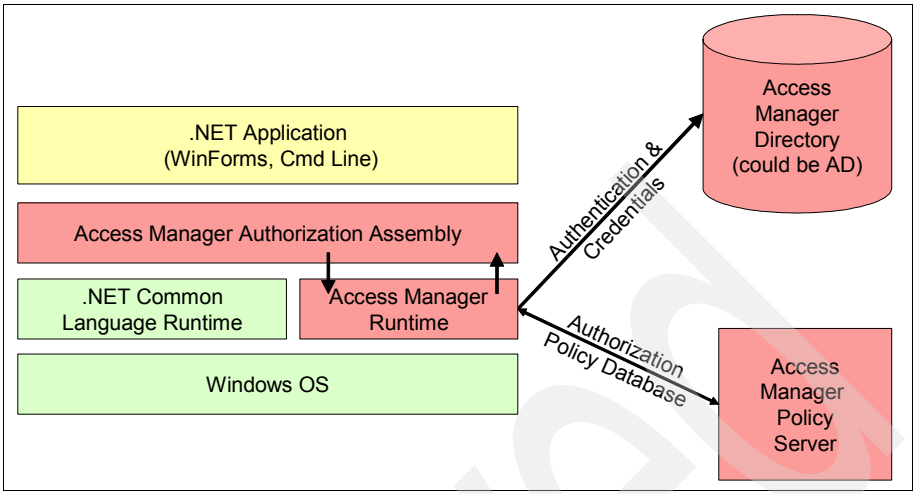


Figure 11-15 Access Manager authorization from .NET application

The Access Manager authorization module contains a locally cached copy of the policy database that is updated by the Policy Server as changes are made by administrative actions that change the Access Manager policy, so authorization checks do not require a call to the Access Manager Policy Server.

Role-Based authorization in ASP.NET

The *Principal to Role mapping* provided with Access Manager for .NET is used by:

- The ASP.NET container to enforce role requirements declared by the application.
- The ASP.NET application that explicitly invokes IsInRole (role-name) to control its logic.

To use *Principal to Role mapping*, the source code of ASP.NET applications using either declarative or programmatic security does not have to be modified in any way.

It is not the intention of this section to explain how to write a .NET application, nor explain security calls, but to explain architectural context. For this reason we are providing an example of C# code to get the current IPrincipal object and check if the user possesses the 'PetOwners' role:

```
IPrincipal currentPrincipal = Thread.CurrentPrincipal;
if (currentPrincipal.IsInRole("PetOwners")) { ...
```

Here is an example from an aspx page:

```
System.Security.Principal.IPrincipal user = HttpContext.Current.User;
if (user.IsInRole("PetOwners")) { ...
```

In both examples, the IPrincipal will be an AccessManagerPrincipal object that was created by the Authentication Module and inserted into the security context of the application. Because AccessManagerPrincipal implements the .NET standard IPrincipal both declarative and programmatic security realms work unchanged.

Figure 11-16 shows the context of role-based authorization in an ASP.NET environment.

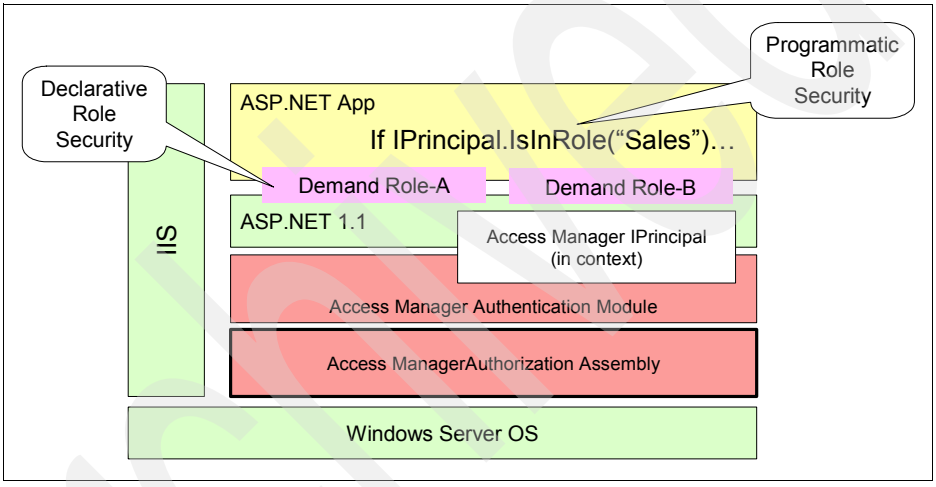


Figure 11-16 Role-based authorization in an ASP.NET environment

Principal to Role mapping approaches

Access Manager for .NET provides two approaches for determining if the user (Access Manager IPrincipal) possesses a given role:

- User-to-Role mapping via the user's group membership (as found in the currently authenticated credential).

  In this case, the roles possessed by a given user are determined by the user's group membership within the current Access Manager credential. That is, a check for role "role name" simply checks if the user's credential contains a group with the same name.

- User-to-Role mapping via an Access Manager authorization check of an object in the Access Manager protected object space that represents the role.

  In this case, roles are represented by objects in the Access Manager object space, and the roles possessed by a user are those role objects that allow the *[AMNET] "m"* permission.

In both cases it is possible to partition the "role name space" – allowing, for example, two applications to use a role called "Finance". This is called the "role context" of the application and is a separately configurable for each application. The role context is a simple string such as "myApp".

In the case of role mapping via group membership the role context is prepended to the requested role with a '/' separator and a check is made for the resulting group name. For example, with a context of "myApp" a check for "Finance" would look for group "*myApp/Finance*". By default there is no role context and the role name equates directly to a group name.

In the case of role mapping via object space permissions the role context is made part of the path to the Access Manager object that represents the role. For example, with a context of "myApp" a check "Finance" would check that the user is granted the *[AMNET] "m"* action on the */AMNET/Roles/myApp/Finance* object. By default there is not a role context and the object space path is */AMNET/Roles/DefaultContext/<role>*.

Figure 11-17 shows both options. Please note that the choice of group or object space role mapping is made for each .NET application.

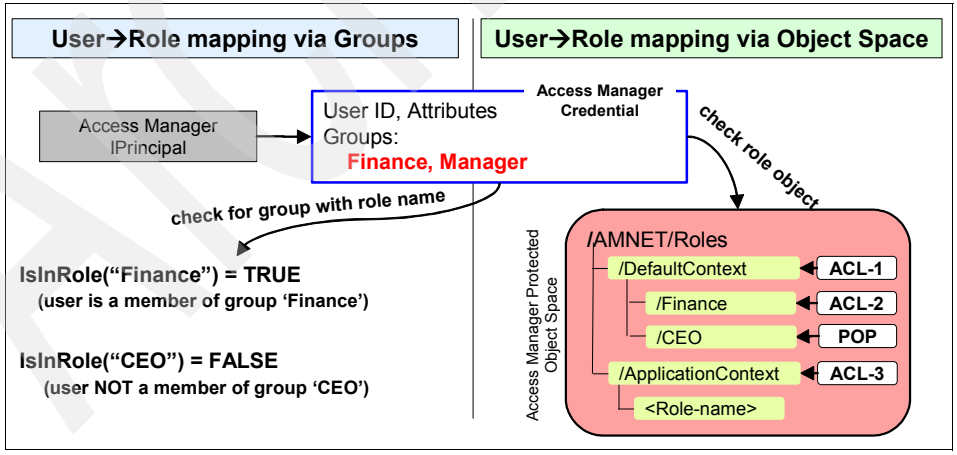


Figure 11-17 User to role mapping: Two distinct configuration options

There are a few considerations that you must be aware of before you decide on one of the models for each application:

- Role mapping via groups
  - Very simple model requires less administration
    - No objects to create, no ACLs to manage
  - Better performance
    - Only need to check Access Manager credential for presence of group (=role)
  - Change of policy not effective until next user login
  - Cannot use advanced authorization policy methods
    - POP, authorization rule
- Role mapping via Access Manager object authorization
  - Allows dynamic policy change without new user login
  - Can use all Access Manager authorization policy declarations
    - ACLs, POPs, authorization rules
  - Good performance (local policy database used for authorization check)
  - More administration setup and Access Manager for .NET configuration

ASP.NET Web services security

Access Manager for .NET provides a plug-in to the MS Web Services Enhancements 2.0 framework (WSE 2.0).

Role-based authorization is available for Web services using either:

- *Transport identity*
  - User is in HTTP header, authenticated by Access Manager Authentication Module
- *Message identity*
  - User is in SOAP WS-Security header, authenticated by Access Manager for .NET plugin for Web Services Enhancements for Microsoft .NET 2.0

This configuration choice is available for each Web service. Please note that *transport* and *message* based authentication can also be used together, and in this case message identity has precedence, if present in a SOAP request.

Message identity can be used for declarative and programmatic role-based authorization. In this case, user-to-role mapping is evaluated by group

membership or object authorization, the same as a typical ASP.NET Web application.

The Access Manager for .NET Plug-in for Web Services Enhancements for Microsoft .NET 2.0 supports:

- Username/Token with Username and Password
- Username/Password validated against Access Manager registry

Each ASP.NET Web service can request that the identity carried in the Username/Token of the SOAP's WS-Security header be used in preference to the identity carried in the HTTP request headers.

When applications use message-level identity:

- Declarative security is evaluated using message-level identity but only if a valid Username/Token with Password is present, and is successfully validated against the Access Manager registry.
- The transport-level identity is not used (for example, the message-level identity overrides the HTTP identity).

The message-level identity can also be used for programmatic security, for example:

```
IPrincipal currentPrincipal = Thread.CurrentPrincipal;
if (currentPrincipal.IsInRole("myRole"))
```

If applications access the message-level identity via the "RequestSoapContext", they will still work and the identity will be the same one accessed via Thread.CurrentPrincipal.

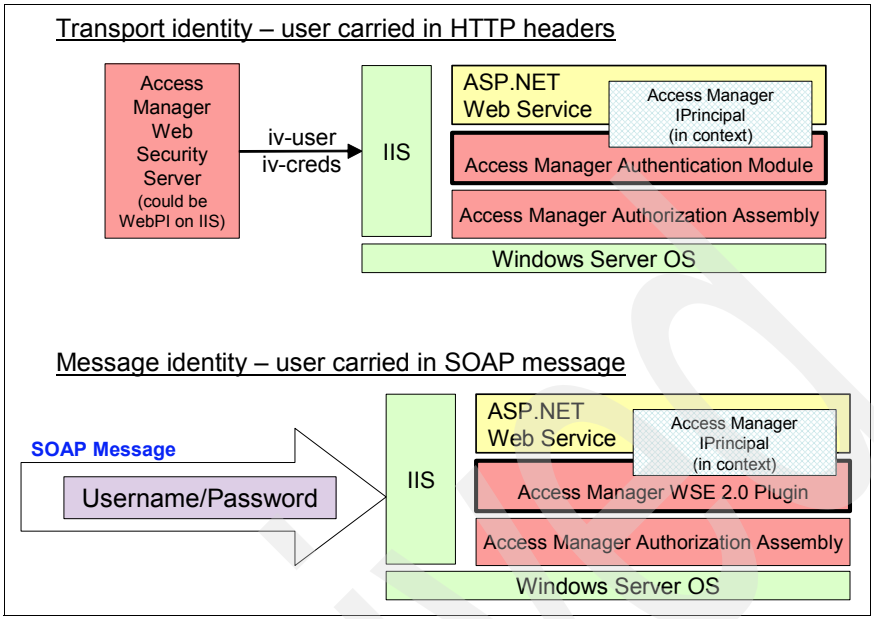


Figure 11-18 HTTP or SOAP security in an ASP.NET Web service

Client side proxy for Web services

The .NET infrastructure and development tools allow a Web service client application to view the Web service as a locally accessible object with an interface that matches the operations of the Web service. That is, the application simply invokes methods on a local object that acts as a proxy for the Web service. The proxy object handles any data format conversion, creation of a SOAP Request package, transporting the SOAP request to the remote service over some transport (usually HTTP) and doing the inverse for the SOAP Response (the .NET support in this regard is analogous to the WSDL2Java development tool and the JAX-RPC support in Java.)

Access Manager for .NET provides a class that integrates into this mechanism to provide client-side Web services security. This class uses the application current authenticated user (in the security context of the application) to:

- Authorize the user's access to the Web service by performing an authorization check on a Access Manager protected object.
- Insert the user's identity into the HTTP request that will carry the SOAP request.

The user can then be authenticated and authorized by the Web service. Note that in order to use this support, a small modification needs to be done at the Web service client application.

In Figure 11-19 we show a .NET application invoking a Web service. Access Manager for .NET authorizes the current user (IPrincipal) access to the Web services. Access Manager for .NET also inserts the user identity into the HTTP headers, which allows user authentication and authorization at the Web service.

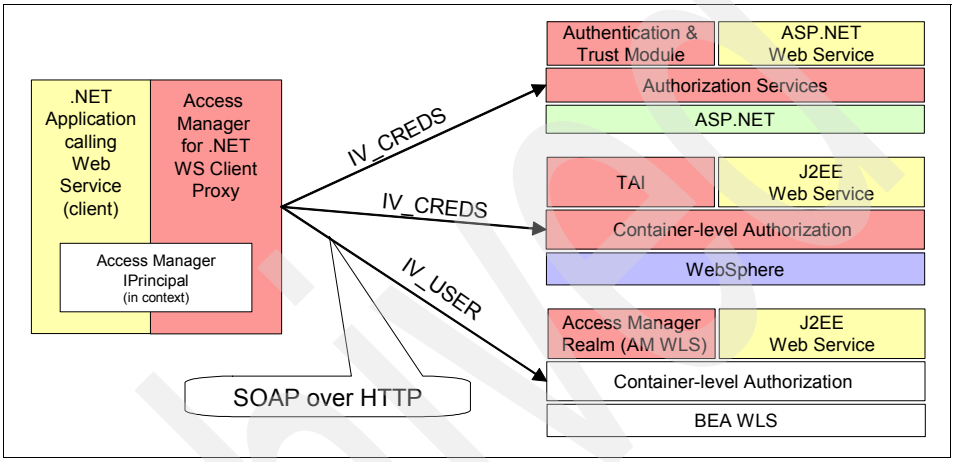


Figure 11-19 Web services client side proxy

11.6 C and Java application integration

If we want to integrate C and Java applications running outside an Application Server, it is necessary to call the Access Manager Authorization API from within this applications.

Access Manager provides a C version of the Authorization API (aznAPI) and pure Java classes: *PDPermission*, *PDPrincipal*, and *PDLoginModule*.

Java wrapper classes for the aznAPI are also available from Open Source. PDPermission is usable in both a Java Authentication and Authorization Services (JAAS) and non-JAAS environment. We provide an overview of these methods in the next subsections, “This interface is called aznAPI. Access Manager provides a C version of the API, and Java wrappers are available as Open Source.” and “PDPermission and JAAS.”

The Authorization API

Using the Tivoli Access Manager authorization application programming interface (API), you can program Tivoli Access Manager applications and third-party applications to query the Tivoli Access Manager authorization service for authorization decisions.

The Tivoli Access Manager authorization API is the interface between the server-based resource manager and the authorization service and provides a standard model for coding authorization requests and decisions. The authorization API lets you make standardized calls to the centrally managed authorization service from any developed application.

The authorization API supports two implementation modes:

- Remote cache mode

  In remote cache mode, you use the authorization API to call the Tivoli Access Manager authorization server, which performs authorization decisions on behalf of the application. The authorization server maintains its own cache of the replica authorization policy database.

- Local cache mode

  In local cache mode, you use the authorization API to download a local replica of the authorization policy database. In this mode, the application can perform all authorization decisions locally.

The authorization API shields you from the complexities of the authorization service mechanism. Issues of management, storage, caching, replication, credentials format, and authentication methods are all hidden behind the authorization API.

The authorization API works independently from the underlying security infrastructure, the credential format, and the evaluating mechanism. The authorization API makes it possible to request an authorization check and get a simple “yes” or “no” recommendation in return.

The authorization API is a component of the Tivoli Access Manager application development kit (ADK).

The Open Group Authorization API standard

The Tivoli Access Manager authorization API implements the Open Group Authorization API (Generic Application Interface for Authorization Frameworks) standard. This interface is based on the International Organization for Standardization (ISO) 10181-3 model for authorization. Figure 10-1 on page 325 explain this model.

This interface is called *aznAPI*. Access Manager provides a C version of the API, and Java wrappers are available as Open Source.

PDPermission and JAAS

The original Java security model dealt almost exclusively with the needs of the Java environment's first major user, the Web browser. It focused on the complexities of secure usage of mobile code, so it worried about the origins of code and its authors, as indicated by digital signatures. The Java 2 environment generalizes that model to concern itself with all code, not just that loaded from remote locations. The Java 2 architecture also restructures the internals of the Java run-time environment to accommodate a very fine-grained usage of security. JAAS, a standard extension of the Java 2 environment, adds in the concept of who the user is who is running the code and factors this information into its security decisions.

All levels of Java security have been policy based. This means that authorization to perform an action is not hard coded into the Java run time or executables. Instead, the Java environment consults policy external to the code to make security decisions, and therefore maps to systems of category 2 or 3, as described previously in 11.2, "Security design objectives" on page 349. In the simplest case, this policy is implemented in a flat file, which somewhat limits its scalability and also adds administrative overhead.

To overcome the flat file implementation of Java 2 policy, and to converge to a single security model, the authorization framework provided by Access Manager can be leveraged from inside a normal Java security check. As mentioned earlier, the most natural and architecturally pleasing implementation of this support is inside a JAAS framework. Support for this standard provides the flexibility for Java developers to leverage fine-grained usage of security and authorization services as an integral component of their application and platform software.

With the Java 2 and JAAS support delivered in Tivoli Access Manager, Java applications can:

- Invoke the JAAS LoginModule supplied by Tivoli Access Manager to acquire authentication and authorization credentials from Access Manager.
- Use the PDPermission class to request authorization decisions.

This offers Java application developers the advantages that:

- The security of Java applications that use PDPermission is managed using the same, consistent model as the rest of the enterprise.
- Java developers do not need to learn anything beyond Java 2 and JAAS.
- Updates to security policy involve Tivoli Access Manager–based administrator actions, rather than any code updates.

Other programming languages

Most programming languages provide some kind of integration with C libraries or Java classes. We can also extend to these platforms to the Access Manager authorization framework.

11.7 Conclusion

Tivoli Access Manager aims to be a corporate authorization solution. Supporting the most important platforms in actual businesses and providing a strong and unique authorization framework across multiple technologies makes Tivoli Access Manager a fundamental enabler of e-business in the B2C and B2B markets.

Access Manager makes application security a reality even with disparate applications that require disparate security approaches, reducing costs of implementing and maintaining proprietary security solutions (islands of security), providing fast time-to-production, reducing cost and complexity of application development, achieving independence through standards and mitigating risks of fraud due to consistent managed end-to-end security.

12

Access Manager for Operating Systems

Compliance and auditing has become a major consideration of corporations. Organizations that use UNIX or Linux operating systems see the benefit of running a robust platform. Yet, these systems have an inherent weakness in producing useful audit information and ensuring compliance with corporate security policy.

This chapter introduces the elements of the Access Manager architecture in a UNIX or Linux environment. It describes the use of the OSSEAL resource manager and covers key architectural issues associated with any Access Manager deployment. It also provides a foundation for the architectural discussions in later chapters.

12.1 Overview of Tivoli Access Manager for Operating Systems

Tivoli Access Manager for Operating Systems provides a layer of authorization policy enforcement in addition to that provided by a native UNIX operating system. It applies fine-grained access controls that restrict or permit access to key system resources. Controls are based on user identity, group membership, the type of operation, time of the day or day of the week, and the accessing application. An administrator can control access to specific file resources, login and network services, and changes of identity. These controls can also be used to manage the execution of administrative procedures and to limit administrative capabilities on a per-user basis. In addition to authorization policy enforcement, mechanisms are provided to verify defined policy and audit authorization decisions.

Access controls are stored in a policy database that is centrally maintained in the IBM Tivoli Access Manager environment. The accessing user definitions are stored in a user registry that is also centrally maintained in the environment. When protected resources are accessed, Tivoli Access Manager for Operating Systems performs an authorization check based on the accessing user's identity, the action, and the resource's access controls to determine whether access should be permitted or denied.

12.1.1 Business context

UNIX has several inherent problems with security when it is examined from an enterprise point of view. One problem is that there is no inherent security infrastructure. Each vendor has their own unique security that can vary widely from platform to platform. Another problem centers around the concept of group users. In UNIX, when a group user account is used such as *root*, all auditing is based on the group user account, not an individual user account. By its nature, this makes auditing events on the host system extremely difficult.

Important: Tivoli Access Manager for Operating Systems *does not* replace native UNIX security. It is, however, an additional level of security.

Tivoli Access Manager for Operating Systems allows a single product to be used to apply security policy consistently across the entire enterprise. The ability to manage an entire corporate UNIX or Linux environment from a single security product has numerous benefits such as:

- Consistent application of security policy regardless of the server
- User-level audit records regardless of whether a group account is used
- Application of a policy from a central server

Figure 12-1 illustrates how Tivoli Access Manager for Operating Systems provides an additional layer of security beyond native UNIX security.

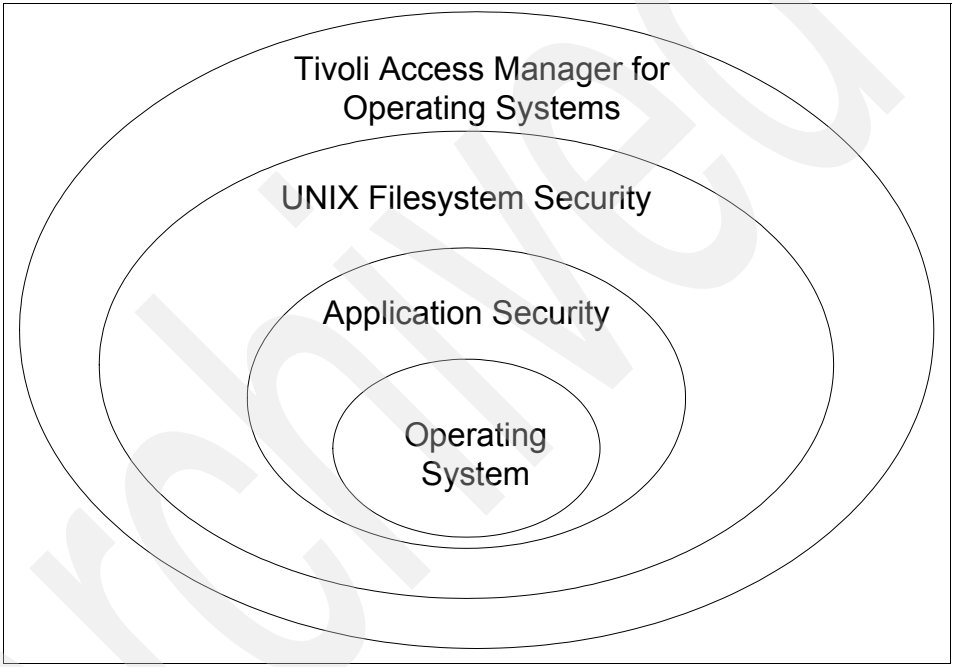


Figure 12-1 Security layering on UNIX with Access Manager for Operating Systems

12.1.2 Access Manager for Operating System integration

Tivoli Access Manager for Operating Systems can leverage the existing infrastructure from other Tivoli Access Manager deployments such as Tivoli Access Manager for e-business. All Tivoli Access Manager systems have the ability to share a common user registry, Policy Server, and Web Portal Manager. Figure 12-2 illustrates this concept.

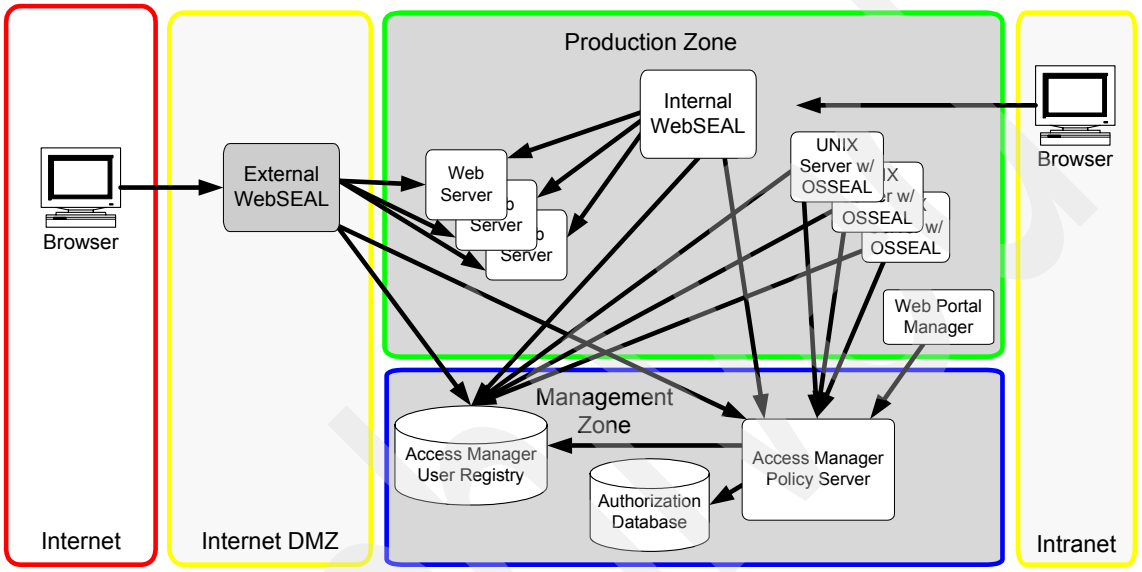


Figure 12-2 Access Manager for Operating Systems and Access Manager for e-business components

For more information, see Chapter 5, “Access Manager core components” on page 163.

12.2 Security architecture subsystems perspective

Architectural subsystems provide a way to group common attributes and to provide a common set of services to a broad range of applications (see 2.1, “Common security architecture subsystems” on page 20). The subsystem approach allows for a clear articulation and understanding of the security solution and enables it to be deployed as a service within a real-world infrastructure.

The main security architecture subsystems addressed by Access Manager for Operating Systems are:

- *Access Control*: Access Manager is used to authenticate users and to enforce security policy at application and system level.
- *Auditing*: The Access Manager components and infrastructure provide a comprehensive logging framework that can be integrated with any threat management system.

Access Manager for Operating Systems uses all of the subsystems, but these two are fundamental to the mapping of Access Manager within an overall enterprise IT architecture.

The design of any architecture must be based on clearly defined and articulated principles that form a foundation for the design process. Whenever in doubt about a design decision, the principles should be used to map a path forward and to justify the overall design.

Here are some key principles that can be applied to an access control solution:

- The security solution must have a central point of authority for security-related information. This authority must support both centralized and distributed management.
  - *Motivation*: This principle drives the need for one source of an authoritative security-related policy within an organization. It enables a consistent policy to be applied across applications and systems, and throughout the organization, while providing a flexible administration framework that fits into and enhances an organization's operation capabilities.
  - *Implication*: This principle implies a high degree of integration, broad coverage, and flexibility required from the products that are chosen to support it. Integration is one of the greatest challenges.
- Security policy should be defined and enforced across all layers of the infrastructure from the application layer down to the network.
  - *Motivation*: The security of any system is only as strong as its weakest link. As a result, it is essential to secure the application, the system on which the applications runs, and the network that supports the solution.
  - *Implication*: Securing all aspects of an IT system always generates numerous integration issues because no one product provides an enterprise security solution. For example, throughout an environment, maintaining policy consistency and consolidation of logging systems are just two of the major issues that must be addressed.

- Sufficient logging is required to capture all authentication and access control decision events and logs. The level of logging should be based on business and security requirements, so the security solution should provide comprehensive and flexible logging coverage, allowing it to be customized.
  - *Motivation*: Because no security solution is foolproof, it is essential to keep good records of the transactions performed by the security system. An easily manageable method of dealing with these records is essential.
  - *Implications*: Strong integration is required to provide logging across multiple systems. Mechanisms must be in place to collect, filter, analyze, and report on audit data.

These principles are not intended to be comprehensive, but to highlight some core objectives of the security solution.

Access Manager for Operating Systems supports all of these principles. The Access Manager family of products, when integrated throughout an environment, provides comprehensive access control capability. The breadth of the Access Manager solution along with its open architecture and interfaces means that it is an optimal solution for providing the majority of an enterprise's access control capabilities.

Access Manager for Operating Systems provides fine-grain access control and audit logging at the system level. While the rest of the Access Manager family of products sits within the application space, Access Manager for Operating Systems sits at the UNIX kernel level to intercept every system call and user transaction. This provides a strong system-level audit capability across a large environment. This capability, in conjunction with the Access Manager application-level logging and the Common Auditing and Reporting Service, can provide a comprehensive operational view of the environment.

12.3 Architecture

Tivoli Access Manager is a network-based access control framework that provides a backbone for defining, managing, and enforcing access control policy. Multiple resource managers can use this framework. Tivoli Access Manager for Operating Systems is one of the resource managers that uses the authorization service provided by Tivoli Access Manager.

Access Manager for Operating Systems uses the same Access Manager base infrastructure as all other resource managers. The core functional components and the base management components are described in the following sections.

Core components

Access Manager is based on two components:

- A user registry
- An Authorization Service, consisting of an authorization database and an authorization engine

Restriction: While Access Manager can use Microsoft Active Directory as a user registry, there are some restrictions when it is used with Access Manager for Operating Systems. Access Manager for Operating Systems can only use a single Active Directory domain.

If the core Access Manager environment is configured into a multiple domain Active Directory environment, then Access Manager for Operating Systems must be configured into only one domain. That domain name must be included in the definitions of access control lists (ACLs), extended attributes, and certain policy objects.

Management components

The Access Manager environment requires certain basic capabilities for administrative control of its functions. Management facilities are provided through the following base components:

- The *Policy Server*, which supports the management of the authorization database and its distribution to Authorization Services
- The *pdadmin utility*, which provides a command line capability for performing administrative functions, such as adding users or groups
- The *Web Portal Manager*, which provides a browser-based capability for performing most of the same functions provided by the **`pdadmin`** utility

For more about the base components, refer to Chapter 5, “Access Manager core components” on page 163.

Although Tivoli Access Manager for Operating Systems relies on the information stored in the centrally maintained Tivoli Access Manager authorization database, the information required to make authorization decisions is replicated and cached within the distributed managed nodes. This enables authorization policy enforcement even if the Tivoli Access Manager Policy Server becomes unavailable.

12.3.1 Authorization model

Tivoli Access Manager for Operating Systems components operate in the user-level application space and within the UNIX kernel. The Tivoli Access Manager for Operating Systems kernel extension and user-level components interact in a tightly integrated, secure manner to provide an extended layer of authorization enforcement as shown in Figure 12-3. Applications access system resources through system-provided APIs, which eventually arrive in the UNIX kernel through a variety of mechanisms.

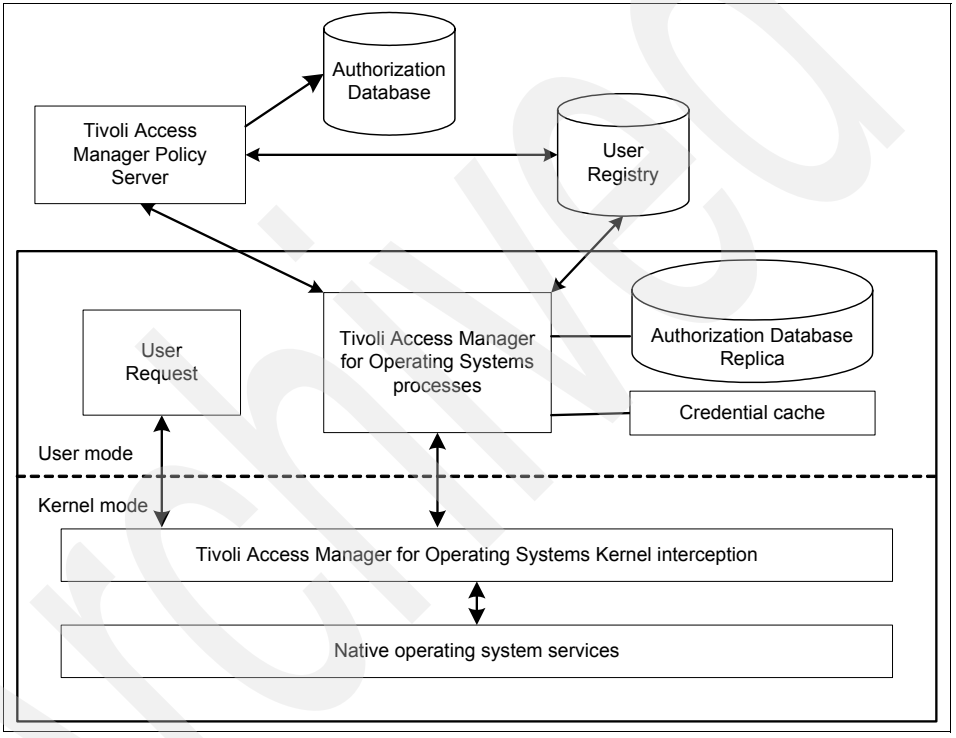


Figure 12-3 An overview of IBM Tivoli Access Manager for Operating Systems

On a system that is not protected by Tivoli Access Manager for Operating Systems, the native system's security verifies whether the accessing user's native identity has the authorization to perform the requested action and either carries out the operation or denies it.

The primary function of the kernel extension is to intervene in accesses to resources that are subject to the authorization policy. The kernel extension uses the authorization daemon process, PDOSD, to obtain an authorization decision and then enforces that decision. If the policy permits access to the resource, the operation continues and is then subject to the native system's security. Otherwise the resource access is denied.

The PDOSD daemon maps UNIX user identities to Tivoli Access Manager credentials that describe users and their group memberships from a Tivoli Access Manager point of view. The PDOSD daemon then uses the Tivoli Access Manager Authorization API to obtain authorization decisions based on the credentials, the operation being performed, the resource being accessed, and its associated access controls defined in the policy database.

12.4 Native UNIX security relationship

One of the most difficult concepts surrounding Access Manager for Operating Systems is how it relates to existing security within UNIX itself. Access Manager for Operating Systems does not replace any existing security. It compliments what is already present on the native operating system. Figure 12-4 on page 390 illustrates how authorization decisions are made on a system running Access Manager for Operating Systems.

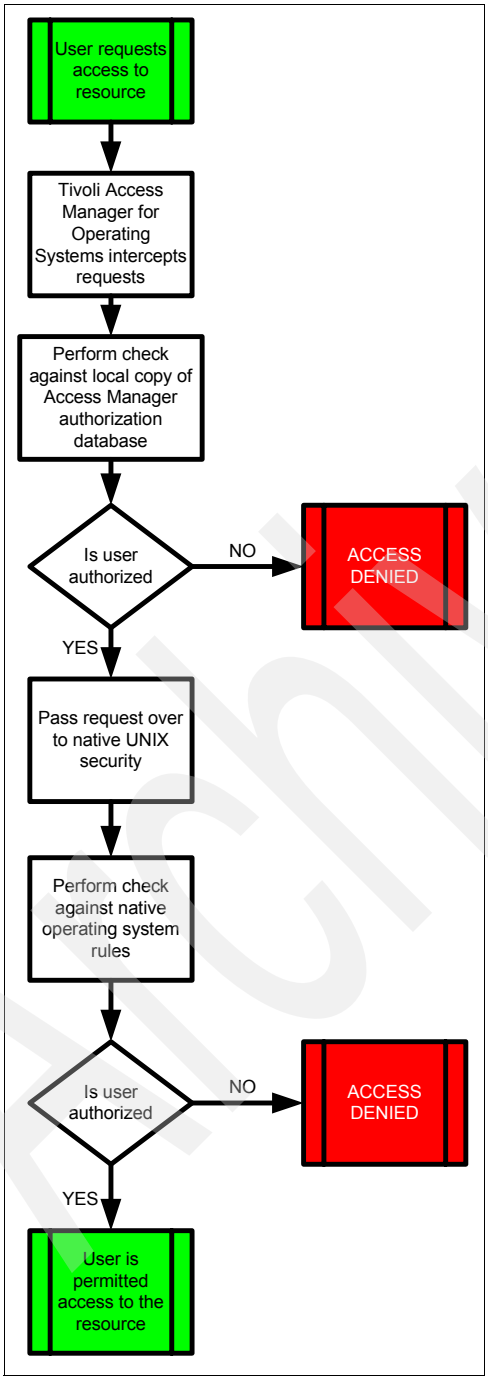


Figure 12-4 Authorization process flow

In addition to the authorization process flows, Tivoli Access Manager for Operating Systems does not replace existing UNIX users. Stated simply, users must exist in UNIX in addition to existing in Access Manager. If a user exists in UNIX and does not exist in Access Manager, the user is treated as an unauthenticated user. This eases implementation issues because the default policy for the enterprise can be applied to the unauthenticated user group within Access Manager. The only users that need to be created in Access Manager are users on UNIX to which the default policy does not apply or is insufficient such as group accounts.

Important: A user that exists within Access Manager's user registry but does not exist on the native UNIX user registry is not allowed to log on to the system. Authentication allows the user to log on to Access Manager services, but it fail when Access Manager for Operating Systems passes the authentication information to the native logon services.

12.5 Policy

IBM Tivoli Access Manager for Operating Systems protects system resources by enforcing authorization policy defined in terms of Tivoli Access Manager access controls. Access to the following types of system resources can be controlled:

- File system resources
- Remote network services
- Local network services
- Login services
- Changes of user and group identity
- Sudo commands
- Password management services

These resources are identified by Tivoli Access Manager object names. They are protected by associating Tivoli Access Manager access controls with the object name. Tivoli Access Manager access controls and object names are also used to specify resource-level and user-level audit policy.

As with all Access Manager resource managers, enforcing access control policy includes the use of:

- *Access control lists*: Identifies specific users, groups of users, and types of users who can be considered for access and specifies the operations permitted on the resource.
- *Protected object policies* (POP): Specifies conditions regarding access to the protected objects, such as auditing, warning mode, and time-of-day access.
- *Extended attributes*: Additional values placed on an object, ACL, or POP that further restrict the access such as limiting which programs can be used to access a resource.

12.5.1 File policy

Tivoli Access Manager for Operating Systems provides the ability to control access to file system resources. File system resources consist of:

- Files
- Directories
- Soft links
- Hard links
- Device files

File system resources are protected in two ways:

- Access controls protect file system resources based on the identity of the user who is attempting the access and the action that the user is trying to perform.
- Membership in the *Trusted Computing Base* (TCB) protects file system resources by monitoring the members' contents and attributes for change.

Table 12-1 details the level of access control that can be applied through the file policy.

Table 12-1 File system permissions

Permission name	Permission granted
Read (r)	Access a file system resource for reading.
Write (w)	Access a file system resource for writing.
Create (N)	Create a particular file system resource.
Execute (x)	Execute a file system resource.
Chown (o)	Change the ownership of a file system resource.

Permission name	Permission granted
Chmod (p)	Change the native UNIX file system permissions associated with a file system resource. This applies to both operations that modify UNIX mode bits and to operations that alter a resource's native ACL for applicable platforms.
Chdir (D)	Change directory into a file system directory resource (directories only).
Rename (R)	Move (or rename) a file system resource.
Delete (d)	Remove a file system resource.
Utime (U)	Modify the file access and modification times associated with a file system resource.
Kill (K)	Terminate a process that was executed from a file system resource.
List (l)	List the contents of a directory.

Trusted Computing Base resources

Tivoli Access Manager for Operating Systems provides the ability to define files on a system as being part of a TCB. Files that are members of the trusted computing base are monitored for changes in ownership, UNIX file permissions, creation and modification time stamps, presence or absence on a system, content of the file, and the device on which the file resides. These attributes are collectively referred to as the *file signature*.

Tivoli Access Manager for Operating Systems has the ability to ignore some of these attributes when creating the file signature. This feature is useful if you want a file to remain part of the TCB even though an attribute has changed. While this reduces the level of security, it allows you to reduce the amount of administrative overhead that needs to occur on file attributes that change frequently. Table 12-2 lists all the attributes that compose the file signature that can be ignored.

Table 12-2 TCB available extended attributes for file signatures

Attribute	Description
Set-CRCMaxFileSize	Set the maximum number of bytes that are considered significant in the calculation of the checksum for the file.
Ignore-CRC	Do not calculate or include the CRC sum in any signature check.
Ignore-CRCExec	Do not calculate or include the CRC sum in signature checks that occur when a program is run.

Attribute	Description
Ignore-Owner	Do not include the user ownership of the file in the signature check.
Ignore-Group	Do not include the group ownership of the file in the signature check.
Ignore-Size	Do not include the file size in the signature check.
Ignore-ctime	Do not include the file creation time in the signature check.
Ignore-mtime	Do not include the last-modified time in the signature check.
Ignore-mode	Do not include the file permission in the signature check.
Ignore-inode	Do not include the inode of the file in the signature check.
Ignore-nlink	Do not include the hard links to the file in the signature check.
Ignore-rdevno	Do not include the device ID of the file in the signature check.
Ignore-devno	Do not include the device number of the file in the signature check.
Ignore-Missing	Do not perform a signature check when the file is not found.
Ignore–All	Do not perform signature checking.

Tivoli Access Manager for Operating Systems enables you to grant special privileges to programs by defining them in the TCB. If the integrity of a program defined in the TCB is compromised, it should no longer be trusted with special privileges. Tivoli Access Manager for Operating Systems detects changes that compromise the integrity of a registered program. When a change is detected, Tivoli Access Manager for Operating Systems records that the program is untrusted and does not allow an untrusted program to be executed until an administrator explicitly trusts it again.

12.5.2 Network policy

Access Manager for Operating Systems provides the ability to control access to remote network services from a local machine. It also provides the ability to control access to local network services from remote locations. These two types of network access are controlled separately by defining protected resources.

Table 12-3 describes the network object names that can be used to define a policy.

Table 12-3 Network resource naming

Object name	Description	Type
protocol	A representation of a network protocol name. The only supported protocol is TCP over IP Version 4 or IP Version 6. This protocol is represented by the string *tcp*.	A case-sensitive string representing the protocol.
service	A description of the set of services represented by this resource. For *NetIncoming* resources, this service represents the service on the local machine to which an incoming connection has been addressed. For *NetOutgoing* resources, this service represents the service on the remote machine to which a connection attempt is being made.	A comma-separated list of ports and port ranges. Ports can be specified explicitly by number or by name. Port names are mapped to port numbers according to the mapping defined in the /etc/services file on the machine where the network policy is being enforced. The special port range '*' is equivalent to the range 1 - 65535. Only one of '*' or '1 - 65535' can be present in your policy.
host	A description of the set of hosts represented by this resource. For *NetIncoming* resources, this represents remote hosts from which an incoming connection is attempted. For *NetOutgoing* resources, this represents the remote host to which an outgoing connection is being attempted.	The host specification may be in one of two forms: ► ip-address[:nbits] ► host name
ip-address	A dotted notation of an IP Version 4 address, for example 192.168.1.42. The notation for IP Version 6 is supported as well.	A string that represents an IP Version 4 or IP Version 6 address.
nbits	The number of bits considered significant in an ip-address. Bits are counted from left to right, with 0 indicating that no bits are significant and 32 indicating that all bits are significant. When a host is specified in the ip-address[:nbits] form and the no nbits component is specified, 32 is assumed.	A number ranging from 0 to 32
hostname	A wildcard string matching the names of the hosts represented by this resource.	A case-insensitive string that consists of wildcard elements and legal host name characters.

12.5.3 Login policy

Access Manager for Operating Systems lets you control when and from where a user can log in to a system. The following basic mechanisms control user access:

- Defining time-of-day login restrictions for users independent from where they log in
- Defining access controls on local and remote terminals

Tivoli Access Manager for Operating Systems also provides the ability to enforce a login-activity-related policy such as password expiry, automatically disabling accounts after a number of failed logins, and automatically disabling inactive accounts.

12.5.4 Password management policy

Access Manager for Operating Systems provides the ability to define and enforce a policy related to password management. Password management prevents users from specifying weak passwords that are vulnerable to compromise by methods such as a dictionary attack. The policy is defined centrally and controls the following aspects of password management activity:

- Password strength
- Password aging

The password management policy is applied in addition to any such policy that is provided natively by the operating system. As a result, the most restrictive policy, which may either be from Tivoli Access Manager for Operating Systems or the operating system, applies.

12.5.5 Surrogate policy

Access Manager for Operating Systems provides the ability to control operations that can change the UNIX identity of a process. Such operations are referred to as *surrogate operations*. Surrogate operations can change the user identity or group identity of a process. Access control of each of these kinds of surrogate operations is established by applying the authorization policy to the User and Group subtypes of the Surrogate resource type. The object names identify the potential targets of the surrogate operations and control the ability, for example, to surrogate to the root user or the system group.

12.5.6 Sudo policy

Sudo resources describe commands that require more stringent access control than whether a particular program can be executed. Sudo commands enable access control based on a command as well as on the parameters that are passed to that command. You can use Sudo commands to remove the requirement for a user to become the root user on a system in order to perform administrative tasks. Sudo does this by providing the capability to run a command as a UNIX user other than that of the invoker.

12.6 Policy branches

Each Tivoli Access Manager for Operating Systems machine is configured to an initial *policy branch* during its initial configuration. The initial policy branch is created and populated with the Tivoli Access Manager for Operating Systems default policy during the initial configuration, if it does not already exist. The branch that is specified as the initial branch should never be a branch that was created by any means other than Tivoli Access Manager for Operating Systems configuration. Doing so can render the machine inoperable. A Tivoli Access Manager administrator can add additional policies to the initial policy branch.

Recommendation: We strongly advise that you do *not* modify the Tivoli Access Manager for Operating Systems default policy.

12.6.1 Single policy branch configuration

Multiple machines can subscribe to the same initial policy branch. This allows an administrator to define the policy for a specific class of machines once and have it apply to all machines that are in that class. If all machines in that class can use the same policy, then any additional policy should be added to the initial policy branch. This is a single branch configuration, which is the easiest type of configuration to understand and maintain.

12.6.2 Multiple policy branch configuration

In some situations, it might be useful to define additional policies in policy branches other than the initial policy branch, such as:

- Each application or set of applications requires their own branch.
- Different hardware types (HP, AIX, Solaris, and so on) can have their own branches.
- Each policy type can have its own branch (File, Sudo, and so on).

Important: Do not configure a second initial policy branch as an additional branch. After the initial policy branch is created during the Access Manager for Operating Systems configuration, there is no reason to create a second initial policy branch (this creates an unnecessary duplicate default policy). Use the defined initial policy branch. If a second branch is required, create it using the appropriate commands either in pdadmin or Web Portal Manager.

For example, application A that runs on a subset of machines. Rather than defining the policy for application A in each of the class-specific initial policy branches, the policy for application A can be defined in its own branch. Then each machine that runs application A can subscribe to this policy branch in addition to the initial policy branch to which it is subscribed. The complexity of understanding the policy that applies to a given machine increases with each additional branch that is configured.

Important: We recommend that you keep the number of configured branches to a minimum.

Figure 12-5 on page 399 illustrates the layout of multiple policy branches for the application A example.

Another problem is also illustrated in Figure 12-5 on page 399. Two conflicting policies are applied to the same resource */etc/passwd*. If an evaluation occurs on the resource, ServerPWD-ACL and AppAPWD-ACL can be applied. To resolve this, Access Manager for Operating Systems uses a method called *branch precedence*. In the example, branch precedence is set to evaluate the AppA branch then the Server branch. For the conflict on */etc/passwd*, it is resolved by using AppAPWD-ACL since AppA has precedence over Servers.

Important: Although branch precedence is used to resolve conflicts, avoid defining conflicting policies. Conflicting policies result in confusion when trying to determine which policy is actually applied.

Consider the following key points, among others, when defining policy and policy branches:

- Defining the multiple policy branches does not negate the need for a proper policy design.
- Policies should be defined to minimize conflicts and maximize security benefits.
- Minimize the number of policy branches whenever possible because complexity grows with each new policy branch created.

- Validate that the correct branch precedence is set for each branch created to avoid applying the incorrect policy when a conflict occurs.

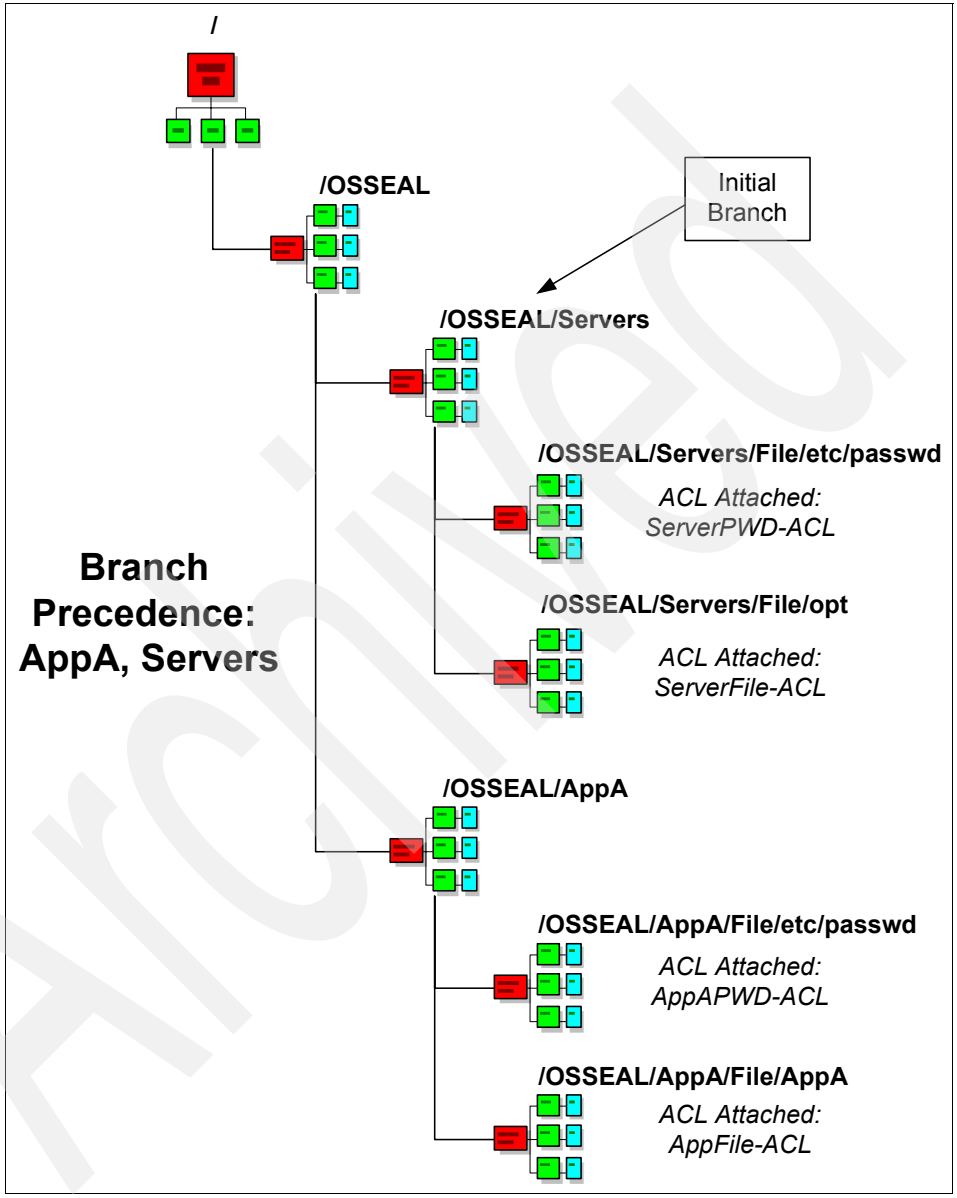


Figure 12-5 Multiple policy branch layout with Access Manager for Operating Systems

12.7 Runtime environment

The following sections describes the major components of Tivoli Access Manager for Operating Systems and their operating environment. The following daemons are responsible for the major functions of Tivoli Access Manager for Operating Systems:

pdosd The authorization daemon makes authorization decisions and monitors the Trusted Computing Base.

pdosauditd The audit daemon receives audit events from other components of Tivoli Access Manager for Operating Systems and manages the audit trail.

pdoswdd The watchdog daemon ensures that the other daemons remain available. The other daemons also monitor each other.

pdostecd The Tivoli Enterprise Console® daemon makes many of the Tivoli Access Manager for Operating Systems audit events available to the Tivoli Enterprise Console.

pdoslpmd The login policy and password management daemon makes authorization decisions about logins and password changes.

pdoslrd The log router daemon makes audit records available for transfer to multiple locations.

Each daemon maintains a log file that records significant events and error conditions. The records written to the log files contain a UTC time stamp, information that identifies the component logging the event, the message classification, and the message text.

12.7.1 The pdosd authorization daemon

The pdosd authorization daemon performs the following actions:

- Handles the authorization requests that are generated by the kernel extension when it intercepts operations that are subject to policy
- Maps UNIX user identities to Tivoli Access Manager credentials that describe users and their group memberships from a Tivoli Access Manager point of view
- Monitors the files that constitute the Trusted Computing Base in order to detect any changes that might cause them to become untrusted

The pdosd daemon is a local-mode Tivoli Access Manager Authorization API application. The Tivoli Access Manager documentation describes this in detail. The pdosd daemon replicates the Access Manager master policy database and

makes authorization decisions based on the information stored in this local replica.

12.7.2 The pdosauditd audit daemon

The pdosauditd audit daemon manages the Tivoli Access Manager for Operating Systems audit log. The audit daemon receives binary audit records from the daemons, kernel extension, and the **pdosobjsig** command. It stores them in memory and writes them to the audit log on a regular basis.

Components generate audit records based on the auditlevel settings. For authorization decisions, the global audit level, global warning level, resource audit level, resource warning level, and per-user audit level are all considered. In the case of a non-authorization decision, only the global audit level is used.

12.7.3 The pdoswdd watchdog daemon

The pdoswdd watchdog daemon monitors the availability of the pdosd, pdosauditd, pdoslpmd, and pdoslrd daemons. These daemons monitor each other in the same manner; this is the watchdog daemon's only function. This self-monitoring function, as implemented by each of the daemons, is the watchdog system. The watchdog system ensures the high availability of Access Manager for Operating Systems services on a machine.

12.7.4 The pdostecd Tivoli Enterprise Console daemon

The pdostecd daemon makes a subset of the audit events produced by Access Manager for Operating Systems available to the Tivoli Enterprise Console. The daemon reads the active log file, /var/pdos/audit/audit.log, and records relevant audit events to a file called /var/pdos/tec/tec.log, which the Tivoli Enterprise Console logfile adapter can monitor.

12.7.5 The pdoslpmd login and password management daemon

The pdoslpmd daemon provides support for Tivoli Access Manager for Operating Systems login activity policy and password management enforcement. Processes that perform logins and password changes communicate with pdoslpmd to determine whether the operation is allowed under the current Tivoli Access Manager for Operating Systems policy.

Login activity and password management policy enforcement is enabled by default when Tivoli Access Manager for Operating Systems is configured on the system. If login policy is not enabled on the system, the pdoslpmd daemon is not running. If login policy is enabled after the initial Tivoli Access Manager for

Operating Systems configuration and start, then the pdoslpmd daemon is started the next time that Tivoli Access Manager for Operating Systems is started.

12.7.6 The pdoslrd log router daemon

The pdoslrd log router daemon reads a Tivoli Access Manager for Operating Systems audit record from an input channel, formats the record, and then queues the record for the output channels to process. Each output channel dequeues a formatted audit record, applies a filter to it (if one has been specified for that channel), and, if the record is not filtered out, formats the record into the proper output format and sends it to its destination.

The pdoslrd daemon uses the audit.log file as input. If the file is removed, the daemon shuts down and must be restarted manually after the audit.log file is made available. The pdosauditd daemon must be shut down and then restarted in order to make the audit.log available.

12.8 Putting it all together

Figure 12-6 shows the component interaction associated with a typical process call within an Access Manager for Operating Systems protected system.

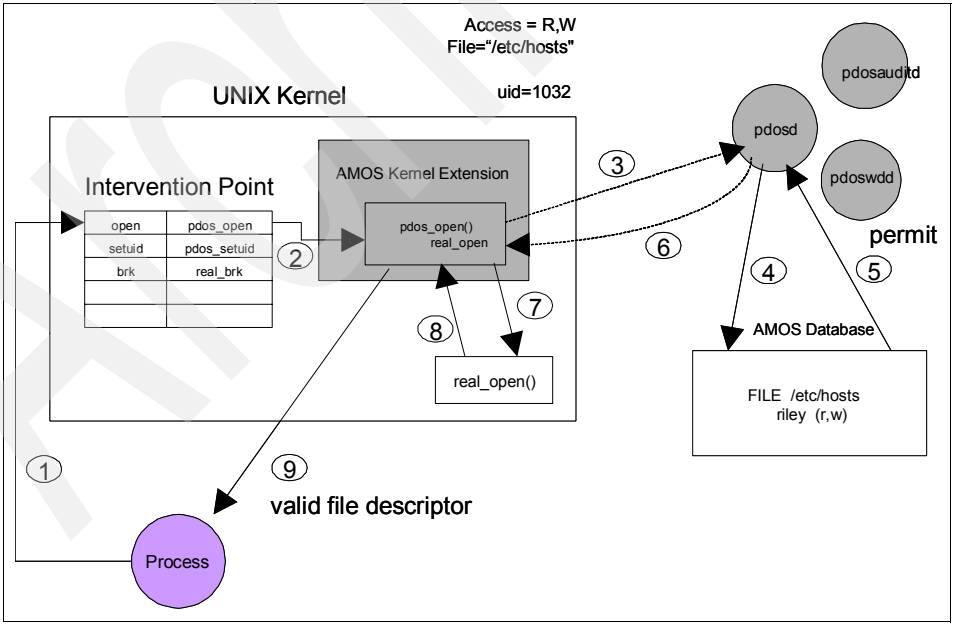


Figure 12-6 Typical Access Manager for Operating Systems component interaction

For each type of operating system and each type of access protection (for example, file system, sockets, and so on), Access Manager for Operating Systems implements an *intervention point* using supported operating system APIs. At the intervention point, the Access Manager for Operating Systems authorization service daisy-chains into the application call process, inserting a call to the Access Manager for Operating Systems kernel-level code. The following data is required to make an access control decision:

- The resource name (for example, /etc/hosts)
- The access mode requested (*read* and *write* in this case)
- The UID of the calling user

In our example, the user logged in as UID 1032 and then performed an **su** command to switch to root (UID 0).

Important: UNIX always keeps track of the UID under which a person originally logged into the system. When a person subsequently uses the **su** command to switch to another ID (or exploits a hacker to reach another ID), UNIX permissions and privileges become based on that new ID. However, Access Manager for Operating Systems always bases access control on the original login ID. This is the most fundamental way in which Access Manager for Operating Systems determines the capabilities of root and protects against root attacks.

Ideally, all administrators log in under a unique user (non-root) ID. Then they use the **su** command to root. Access Manager for Operating Systems checks their original ID first to determine whether the policy allows the operation. If it does, then the operation is passed to UNIX to allow them to perform privileged operations. Applications are otherwise unaffected, and operation methods and processes continue as normal. No special processes or tools are required by the administrators.

The following steps correspond to the numbers in Figure 12-6 on page 402:

1. The executable causes a file system open file call to be made.
2. Access Manager for Operating Systems kernel code checks whether this call is subject to policy. If not, the original call is passed to the operating system routine. If the policy is cached locally and the request is denied, then a standard operating system *no access* return code to the call is immediately returned. If the policy is cached and the request is allowed, the request is immediately passed on to the original operating system routine (and the process skips to step 7. on page 404).
3. If the policy is not cached in the kernel, then the call is passed to the user-level pdosd daemon.

4. pdosd resolves the request from policy data held in a store in the same UNIX system (includes caching).
5. The result is posted back to pdosd.
6. pdosd passes the result back to the Access Manager for Operating Systems kernel-level code.
7. If access was granted, the call is passed on to the original operating system routine.
8. and 9. The file handle from the original operating system routine is returned to the application.

All of these operations take place independently of the Policy Server on which the authorization database master is stored. All of the data needed to make the access determination has already been cached locally. Connection to the Policy Server occurs only if the policy changes and the Policy Server informs the Access Manager for Operating Systems system that it must request a refresh of the access control data cache.

12.9 Entitlement reports

It has traditionally been difficult to determine exactly the access permissions, or *entitlements*, that a given user or group has. The multiple policy branch feature makes this even more difficult.

Tivoli Access Manager for Operating Systems allows for an administrator to quickly determine exactly what a user or group has permissions to do through the use of *entitlement reports*. The **pdosent** command provides the capability to create an entitlement report for a given user or group. This entitlement report contains all information stored in the Tivoli Access Manager Policy Server about the user or group and all the policy information specific to the specified branch precedence rule.

Here are some examples of information that entitlement reports can provide:

- Show entitlements for a user on the present machine
- Show entitlements for members of a given group
- Show entitlements for a user if the machine was configured to a different domain
- Show entitlements for a given user to perform Tivoli Access Manager base actions

More complex entitlement reports can be generated to see the entitlements that a user or group has if the branch precedence order was changed. Consider the following example with the group *sysadmin*:

```
pdosent -g sysadmin -b branchC,branchB,branchA
```

The command creates an entitlement report for the sysadmin group if the branch precedence was changed to branchC, branchB, and branchA.

Using entitlement reports allows an administrator to not only examine permissions for an existing configuration, but to also see the permissions that are applied if a configuration is altered.

12.10 Auditing

Tivoli Access Manager for Operating Systems provides extensive auditing capabilities. They enable you to track authorization access decisions made to protected resources and to monitor activity of an administrative nature, such as the starting and stopping of the daemons.

Note: From a security architecture subsystem perspective, the core functionality of Access Manager for Operating Systems also supports the Audit subsystem. Access Manager for Operating Systems provides strong auditing capabilities at a granular level. It also supports a consolidated and centralized view of these logs.

12.10.1 Auditing authorization decisions

Auditing of authorization decisions can be set globally, for a specific protected resource, or on a per-user basis.

It is possible to audit authorization access decisions for specific resources by enabling resource-level auditing. This is achieved through POP access controls by setting the audit-level attribute to permit, deny, or both.

Permit Logs all permitted actions

Deny Logs all denied actions

Both Logs all action on the resource

Audit records for authorization access decisions are also generated if the permit or deny level is set in the global audit level. This results in the generation of audit records for all authorization decisions that permit or deny access to protected resources. The global audit level is set on a per-machine basis.

Note: The auditing levels for the global audit level and the resource audit level are cumulative. For example, if the global audit level is set to `deny`, and a resource has a POP attached to it with an audit level of `permit`, every authorization decision for access to that resource is audited.

It is possible to enable more granular auditing for authorization access decisions based on the action that is being performed against the protected resource. This granularity can be accomplished by specifying the accessing permissions that trigger the generation of an audit record. This action can be useful in reducing the total amount of audit records that are generated.

For example, for file system resources, it may be desirable to only audit authorization decisions that permit actions that could modify the file resource, such as the actions kill program (K), create (N), rename (R), delete (d), change ownership (o), change permission (p), and write (w). Auditing based on the action that is being performed can be specified separately for global permit, global deny, per-resource permit, and per-resource deny audit levels. It only has an effect if the corresponding global or per-resource audit level is enabled.

Login audit

It is possible to audit authorization decisions that are specific to login by setting the global *loginpermit* and *logindeny* audit levels. These generate audit log records for authorization decisions that permit or deny a login action respectively. Authorization decisions that are specific to login are also audited if the global permit and deny audit levels are set. The loginpermit and logindeny audit levels enable global audit of login separately from other authorization decisions.

Audit authorization decisions on a per-user basis can also be defined by the user-level audit authorization policy using AuditAuth resource definitions. The user-level audit authorization policy can be set on an individual user, a group, or unauthenticated users. The supported audit levels for user-level audit authorization policy are permit, deny, loginpermit, logindeny, all, and none.

The permit and deny audit levels enable the generation of audit records for all authorization decisions that permit and deny access by the user to protected resources.

The loginpermit and the logindeny audit levels enable the generation of audit records for all login-related authorization decisions that permit and deny a login by a user. Specifying `all` turns on all of the audit levels. A level of *none* is a special case that indicates that no audit records should be generated for the user even if global or resource level auditing is set. With the exception of audit level none, all audit levels are additional to the audit levels set by global and resource audit.

12.10.2 Auditing administrative activity

You can also audit actions taken by the Access Manager for Operating Systems daemons of an administrative nature by setting the administrative level in the global audit level. The administrative audit level causes Tivoli Access Manager for Operating Systems daemons to generate audit records for the following events among others:

- Starting and stopping the daemons
- Loss of connectivity with the Tivoli Access Manager user registry
- Trusted-Computing-Base-related activity such as a file being marked untrusted by the Trusted Computing Base monitoring function
- Detection of the incorrect policy

The administrative audit level also causes the generation of audit records for events that are related to a user login account being enabled or disabled when login activity policy is being enforced. Enabling the information level in the global audit level causes auditing to occur for routine events such as the pdosd daemon receiving valid policy updates. Setting the information level results in a large amount of audit data being generated.

Recommendation: Carefully monitor the overall space that is consumed by audit data. We recommend that you place audit logs into a separate file system so that they do not fill up the root file system.

12.10.3 Auditing trace events

Tivoli Access Manager for Operating Systems supports the generation of TraceExec and TraceFile audit events. Trace-style audit events are generated by setting the trace_exec, trace_exec_l, trace_exec_root, or trace_file levels in the global audit level or defining user-level trace policy.

Setting the trace_exec global audit level causes the Tivoli Access Manager for Operating Systems kernel code to track program invocations initiated by the exec() system call that occurs in processes that descend from a login event that was detected by Tivoli Access Manager for Operating Systems. This action results in the generation of a TraceExec audit record for each detected exec() system call. These records are generated regardless of whether the program being run is protected by the Tivoli Access Manager for Operating Systems policy. Depending on the amount of activity on the system, activating the trace_exec global audit level can generate a large amount of auditing data that can be difficult to manage.

When the trace_exec_l global audit level is enabled, and the trace_exec audit level is not enabled, TraceExec audit data is only generated for the exec() activity when the accessing user's accessor identity and effective UNIX identity do not match. This typically happens when a user surrogates to another user. Use of the trace_exec_l level, instead of trace_exec, prevents TraceExec audit data from being generated when the accessing user's accessor identity and effective UNIX identity are the same.

When the trace_exec_root global audit level is enabled, and the trace_exec audit level is not enabled, TraceExec audit data is only generated for the exec() activity when the accessing user's accessor identity is the root user. Note that it is the accessing user's accessor identity, the identity used by Tivoli Access Manager for Operating Systems for purposes of making authorization decisions, that matters, not the user's effective UNIX identity.

Using both the trace_exec_root and trace_exec_l audit levels, instead of trace_exec, causes TraceExec audit data to be generated only for program invocations that are initiated by the exec() system call. It occurs in processes that descend from a login event that was detected by Tivoli Access Manager for Operating Systems. It also occurs when the accessing user's accessor identity is either the root user or the accessor identity and the effective UNIX identity do not match.

Setting the trace_file global audit level results in the generation of a TraceFile audit record for each access to a file system resource that is protected by Tivoli Access Manager for Operating Systems policy.

12.10.4 Audit log consolidation

Tivoli Access Manager for Operating Systems supports the functionality to send audit data to the following three destinations: a local text file, an e-mail address, and a remote collection point, which is a Tivoli Access Manager authorization server, pdacld. It also supports the functionality to send audit data to all three destinations. The data that is sent to these destinations can be filtered and formatted.

The audit log consolidation functionality is controlled by pdoslrd, the log router daemon. The daemon reads a Tivoli Access Manager for Operating Systems audit record from an input channel (the audit logs), formats the record, and sends the formatted record to the appropriate destination (local file, e-mail, or remote host). A control file is used to specify the destination channels and associated filters.

Multiple Tivoli Access Manager for Operating Systems machines can be configured to send audit data to the same pdacld server as the remote collection

point, consolidating the audit data into a single file. A command line utility, **pdoscollview**, is provided to view the audit records stored in the consolidated audit log file.

12.10.5 Common Auditing and Reporting Service integration

Tivoli Access Manager for Operating Systems has the ability to use the Common Auditing and Reporting Service for consolidating and centralizing audit log information.

Common Auditing and Reporting Service

The Common Auditing and Reporting Service has the following features:

- Provides auditing support
  - Defines a consistent format for auditable events using the Common Base Event (CBE) format
  - Provides a centralized collection point for auditable events from various sources
  - Provides consistent management of the lifecycle of audit data
- Facilitates reporting of audit data
  - Provides interfaces to stage audit data into custom report tables
  - Enables customers to use a reporting tool of their choice to build custom audit reports
  - Facilitates cross-product audit reports
  - Exploits IBM products to provide audit reports for immediate use
- Provides interfaces for IBM products to create and submit data that needs to be audited

For more information about the Common Auditing and Reporting System, refer to Chapter 27, “Introducing IBM Tivoli Common Auditing and Reporting Service” on page 845.

Log file consolidation

By using the Common Auditing and Reporting Service, the numerous audit log files generated by Tivoli Access Manager for Operating Systems can be consolidated into a central repository for analysis. Several reports are shipped with the product and are summarized in Table 12-4.

Table 12-4 Access Manager for Operating Systems audit reports

Report name	Description
General Audit Event Details Report	Displays all information about a single auditable event denoted by the event reference ID parameter. Typically a user runs this report after running other reports and deciding that an event drill down is desired.
General Audit Event History	Displays the total number of auditable events for each event type during a specified time period. It also shows all events of the specified event type and product name sorted by specified sort criterion and time stamp. This report can be used for incident investigation and assuring compliance.
Audit Event History by User	Displays the total number of events for a specified user during a specified time period. It also presents a list of all events of the specified event type and product name sorted by time stamp and grouped by session ID during the time period. The purpose of this report is to investigate the activity of a particular user during a specified time period.
Failed Authentication History	Presents a list of all failed authentication events over the time period sorted by specified sort criteria such as by time stamp. This report can be used by an administrator to investigate security incidents.
Failed Authorization History	Lists all of the failed authorizations events during a specified time frame.
Locked Account History	Displays all of the accounts that have been locked during a specified time period.
User Password Change History	Displays events that are related to password changes done by the user during a specified time period.
Server Availability Report	Shows the availability status of Security servers on a specific machine. The user can display all protected machines in the report or limit the report by entering a single host name as the subject of the report.

Report name	Description
Certificate Expiration Report	Allows detection of soon-to-expire certificates and highlights the need to replace the certificate to insure 24x7 operability. It shows the number of clients that have server or Secure Sockets Layer (SSL) certificates that expire in *n* days. It also show a table of client host names, the days until their certificates expire, and the server to which they are configured.
Most Active Accessors Report	Shows a list of users who are the most active in the system, and can lead administrators to investigate improper use of their resources.
General Authorization Event History	Displays the total number of authorization events, failed authorization events, successful authorization events, and unauthenticated events during the specified time period. Additionally it shows a list of all authorization events sorted by specified sort criteria (time stamp, resource, or user name) during the time period. The purpose of this report is to analyze authorization event history for incident investigation and assuring compliance.
Authorization Event History by Action	Displays a list of all authorization events that contain the specified action sorted by resource and then time stamp during the time period specified.
User Administration Event History	Can be used to investigate security incidents and to track changes to users by administrators.
Resource Access By Accessor Report	Shows the top resources in terms of access or authorization events during a time period for each machine name identified. The report identifies who is repeatedly accessing resources and the resource that is being accessed.
Resource Access By Resource Report	Shows the top accessors in terms of access or authorization events during a time period for each machine name identified. The report identifies which resources are most heavily accessed and which user is accessing the resource.

12.11 Conclusion

This concludes the discussion about the Tivoli Access Manager for Operating Systems business context, architecture, and technical components. In Chapter 13, “Access Manager for Operating Systems business scenario” on page 413, we explore some real-life customer scenarios.

13

Access Manager for Operating Systems business scenario

This chapter uses the Stocks-4u.com example that is described in previous chapters. The requirements are based on the requirements defined in Chapter 7, “A basic WebSEAL scenario” on page 245. The business requirements are mapped to system-level technical security requirements.

13.1 Business requirements

To meet regulatory compliance and internal security requirements, Stocks-4u.com has established that implementing IBM Tivoli Access Manager for Operating Systems will mitigate security risks that currently exist in their UNIX environments. They have also decided that current auditing capabilities on UNIX are insufficient to provide the level of detail required by new legislation governing Stocks4u.com's industry.

Due to the wide variety of UNIX vendors in their environment, Stocks-4u.com has decided that a single policy enforcement engine is necessary. This ensures that a single, centralized view of security policy for the entire UNIX-based server environment is available to both administrators and management. The system must be able to have policy defined:

- Globally for all UNIX servers
- On a per vendor basis (for example, IBM AIX)

The corporate direction is to use the new security system to ensure that UNIX-based systems now have the ability to define policy according to the corporate security direction. Management will not allow complete root-level authority to be given to anyone without an audit trail. The audit trail must indicate the actual user performing the operations in order to be compliant with internal and external security requirements.

Stocks-4u.com has indicated that security management should not be a apart of a UNIX administrator's job and, therefore, wants to transfer security administration to their enterprise security group.

13.2 Functional requirements

We extract functional requirements by mapping business requirements to their underlying reasons. We expand the reasons in increasing detail until we find problems that can be solved using capabilities of Access Manager for Operating Systems. Our functional requirements will tie these low level reasons for a business requirement to the Access Manager for Operating Systems capability that can fulfill that business requirement.

Let us examine the business requirements and search for reasons and the functional requirements.

- Business requirement: Single policy enforcement engine.

  Stocks4u.com has multiple UNIX vendors in its environment. Each one has a security policy that is defined locally to individual servers. Because Stocks4u.com has specialists for each vendor that supports their UNIX environment, the security policy is not consistent from vendor to vendor or from server to server. This leads to the functional requirements listed in Table 13-1.

  Table 13-1 Functional requirements for security policy enforcement

Requirement	Description
1	All UNIX server security must be implemented from a central location.
2	The security policy must be implemented at a global level and at a vendor level, such as IBM AIX.

- Business requirement: Single view for UNIX security policy.

  Stocks4u.com has never been able to have a complete view of its security policy for their UNIX server environment. This is due to the many tools that exist per platform and the various implementations that exist. Stocks4u.com must have a complete, unified view of security policy across all UNIX servers. This leads to the functional requirements listed in Table 13-2.

  Table 13-2 Functional requirements for security policy management

Requirement	Description
3	All platform and system accounts must be managed centrally.
4	A common Web Interface will be used for security policy management.

- Business requirement: The ability to audit the root user account.

  The biggest problem when a UNIX system audits root user actions is that it is difficult to tie those operations back to a specific user. Stocks4u.com must be able to tie operations performed on their UNIX servers back to a specific user account to comply with internal and external regulations. This leads to the functional requirement listed in Table 13-3.

  Table 13-3 Functional requirement for root user account auditing

Requirement	Description
5	All root operations must have the original user account as part of the audit record that is generated.

- Business requirement: UNIX security should be externalized from the UNIX system administration group.

  Stocks4u.com wants to transfer all security management responsibilities to their security group. This ensures that security policies are implemented correctly across their entire UNIX server environment. This also enables their UNIX system administrators to focus on technical UNIX issues instead of being bogged down in security policy definition and enforcement. This leads to the functional requirement listed in Table 13-4.

Table 13-4 Functional requirement for externalizing UNIX security

Requirement	Description
6	Security policy definition and enforcement will be externalized from the UNIX system administration group.

13.3 Designing the solution

To effectively design a solution, it is important to consider the following key questions:

- What are the various UNIX vendors that are currently implemented?
- Can the solution support every vendor?
- Are there common policies that can be defined globally for the entire environment?
- Is there a logical separation of policy definition where a subset of policy needs to be enforced?
- Who defines the policy?
- What level of auditing is required to meet corporate and regulatory requirements?

After the business and functional requirements are defined, you can look at the specific security design objectives. The security objectives and the subsystems become the basis for the conceptual architecture and the implementation phase. The security phase should include security policy definition, policy management, audit management, as well as all standards, guidelines, and policies that relate to UNIX operations.

The requirements for this access control subsystem are typical for those found in many Web application environments. The application-level security is addressed by the use of Tivoli Access Manager for e-business and WebSEAL as described in previous chapters. However, a gap exists in the security controls placed on the systems that support the Web applications. That is, in most environments, the system security policy is applied on a per-server basis. Any form of automation

and consolidation of operational functions, such as policy audit and system and security event alerting, requires a substantial amount of custom, in-house scripting and development. This leads to numerous inconsistencies and integration issues within an environment.

To address this, Access Manager for Operating Systems is deployed as an additional layer of security at the UNIX system level. The Access Manager for Operating Systems deployment is integrated into the existing Tivoli Access Manager infrastructure and provides centralized policy enforcement and audit capabilities for the whole Stocks-4u Web environment.

Stocks4u.com has decided to implement IBM Tivoli Access Manager for Operating Systems to meet their business requirements. The solution provides the following support:

- Provides a consistent security policy across all Stock-4u.com servers
- Enables a security policy to support the network zones security classification requirements

  That is, the security policy is most secure and locked down in the Internet DMZ, and more open within the more trusted zones.
- Provides a framework to enable the rapid deployment of the new server infrastructure
- Protects application-level transactions down to a data storage level
- Minimizes data flow between the Stocks-4u sites in Savannah and San Diego
- Provides a consistent audit and logging framework, with centralized log consolidation capabilities for event alerting and auditing
- Supports system administration from multiple locations in a consistent manner

13.4 Policy design

The main considerations when defining the system security policies are:

- User and administration access

  Which users are allowed to manage the servers and under what conditions can this be done?
- Available services for each type of server

  Which connections does a server accept and which connections are allowed to other servers?

- Logging requirements

  What needs to be logged and to what level of detail does it need to be logged?

- Operational monitoring

  What other monitoring needs to occur, such as snmp alerting, syslogs, process and CPU monitoring?

13.4.1 Administrative groups

For this example, we have four types of administrators within the Stocks-4u Web environment:

- Network security

  This group manages the firewalls and Internet DMZ infrastructure, including WebSEAL servers, mail, and Domain Name System (DNS). This group also handles the Access Manager for Operating Systems administration activities.

- Tivoli Access Manager for e-business administrators

  This group manages the Tivoli Access Manager for e-business application and deployment within the Stocks-4u environment. This includes any version upgrades and configuration changes that are required.

- Application managers

  This group manages the business applications.

- Security administrators

  This group manages user access and user provisioning. It is responsible for creating users and placing them in appropriate groups and so on.

13.4.2 Policy layout

To meet Stocks4u.com's need for global and vendor specific policies, the policy tree within Access Manager for Operating Systems must be defined appropriately. Table 13-1 on page 415 illustrates the appropriate policy branch definition for Stocks4u.com.

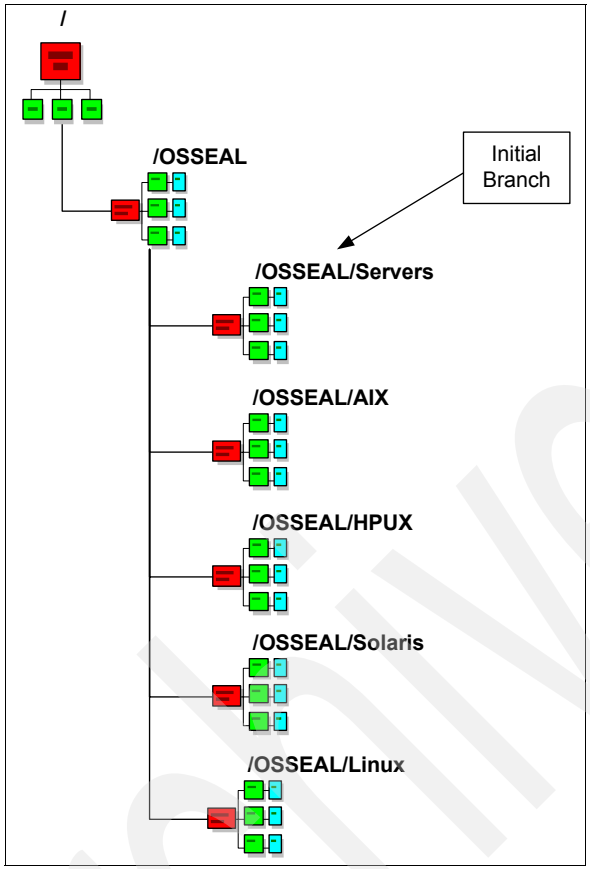


Figure 13-1 Stocks4u.com policy branch definition

The policy branch definition allows for all servers in the Stocks4u.com environment to subscribe to the */OSSEAL/Servers* branch for global policy. In addition, each specific operating system (AIX, HP-UX, Solaris, Linux) can subscribe to their own branch. Branch precedence is defined as the specific operating system branch first and then the global server branch.

The security management department at Stocks4u.com can now implement a policy globally for all UNIX-based servers as well as apply a specific policy to servers that run any vendor's UNIX operating system.

We further examine the specific policies to be applied to the AIX servers at Stocks4u.com. Because Stocks4u.com runs all of the Access Manager base components (Policy Server, Authorization Server, Policy Proxy Server, Common Auditing and Reporting Service, and User Registry) on AIX servers, there are

specific security requirements that must be defined in the /OSSEAL/AIX branch. This differs from the Mail and DNS servers that run on HP-UX.

AIX servers

The groups that allowed access to these servers are the network security group and the Tivoli Access Manager for e-business administrators. Each user has a unique login, and remote root login is not allowed. Tivoli Access Manager for e-business administrators have access only to Tivoli Access Manager for e-business directories and files. The network security group has access to the whole system.

The *external interface* allows connections from any source on port 80 and 443 only. The *internal interface* allows outbound connections to the Lightweight Directory Access Protocol (LDAP) replica and Web servers in the production network on their respective ports only.

The management interface accepts Tivoli Access Manager management connections from the Policy Server and the Policy Proxy Server. It also accepts Secure Shell (SSH) connections from the system management server within the management network for CLI administration. In addition, it allows outbound connections for systems management traffic only to specific servers.

A Trusted Computing Base (TCB) is established within the servers to protect core system files such as routing tables, password files, and so on.

HP-UX servers

The HP-UX servers are solely managed by the network services group, who has full access to the servers. This group is responsible for running the mail and DNS applications for the company.

Inbound connections are allowed to UDP port 53 for DNS and TCP port 25 for Simple Network Management Protocol (SNMP) on the external network interface. *Outbound connections* are allowed for SNMP to the actual mail servers within the internal network.

The management network allows SSH connections from the systems management servers. It also allows the same outbound connections as for the WebSEAL servers.

A TCB is established within the servers to protect core system files such as routing tables, DNS tables, password files, and so on.

13.4.3 Architecture overview

An environment that does not have any pre-existing Access Manager components installed is illustrated from a logical viewpoint in Figure 13-2.

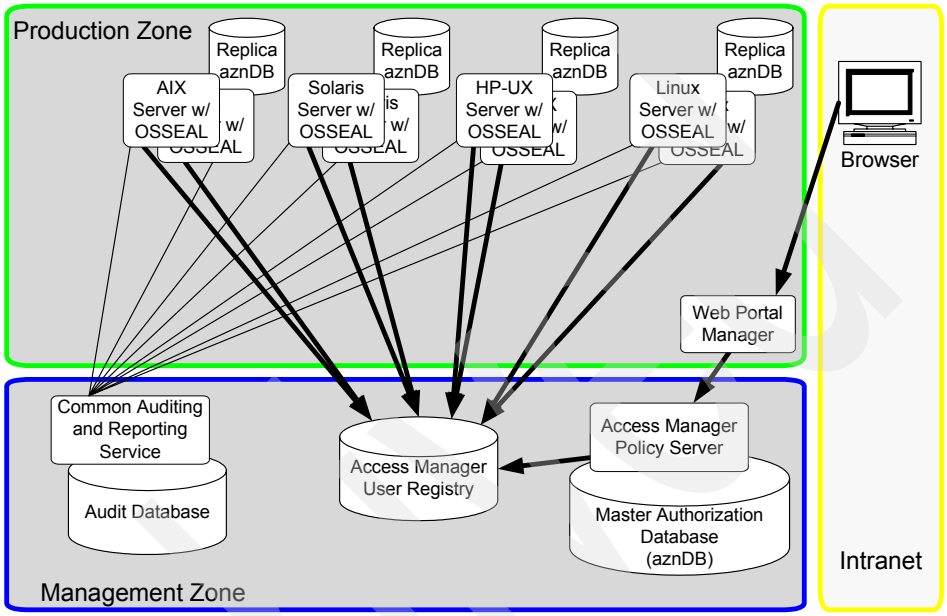


Figure 13-2 Access Manager for Operating System logical component placement

Each server secured by Access Manager for Operating Systems shares the same common requirements:

- Each server runs the Access Manager for Operating Systems daemon (OSSEAL).
- For authentication purposes, each server must communicate with the Access Manager user registry.
- Each server has a local copy of the authorization database (aznDB) to evaluate security policy decisions.
- Every server sends the audit data to the Common Auditing and Reporting Service for audit consolidation and reporting.

13.5 Integrating into an Access Manager environment

Since Stocks4u.com already has an existing Access Manager environment deployment to secure their Web applications, the corporation wants to leverage their existing environment wherever feasible.

Figure 13-3 shows the environment into which Access Manager for Operating Systems is deployed. The view is given at a logical network level to illustrate the network zones that dictate the security policies applied to the servers within them.

The system security policy for each server is driven by two main factors.

- Server function, such as Web servers, mail servers, and application servers
- Server placement, including Internet DMZ, intranet, management zone, and so on

The environment in Figure 13-3 depicts the San Diego location.

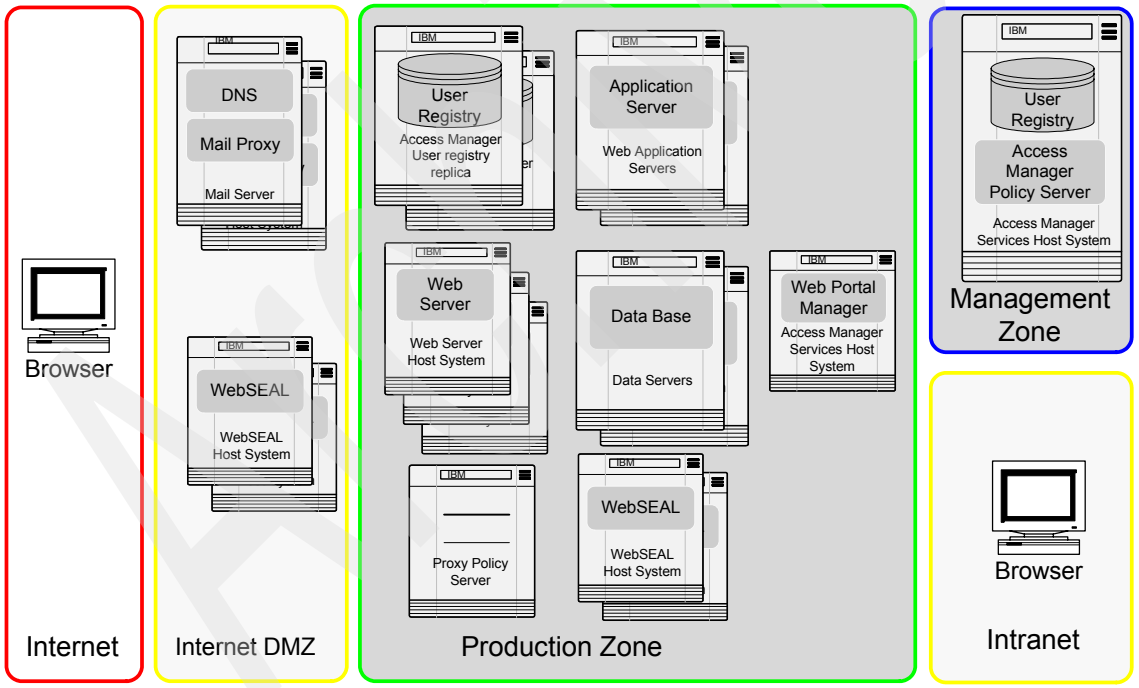


Figure 13-3 Servers deployed within the Stocks-4u Web environment

To minimize the disruption to the existing Access Manager environment at Stocks4u.com, a new Access Manager domain called *amos* is created strictly for use by Access Manager for Operating Systems. This ensures that security

policies for the Web environment remain separate from the security policies of the UNIX environment.

Important: When you integrate multiple resource managers, such as Access Manager for Operating System and Access Manager WebSEAL, consider the amount of policy replication traffic that may occur. If the amount of policy replication traffic will be large for one environment and small for the other, it is better to split the environment into two distinct Access Manager domains to minimize replication traffic. Each security domain has its own ACL database. Therefore, an environment with multiple domains will only replicate the security on a per-domain basis, and not to the whole environment.

Figure 13-4 illustrates the placement of Access Manager components when integrating with an already existing environment such as Tivoli Access Manager for e-business.

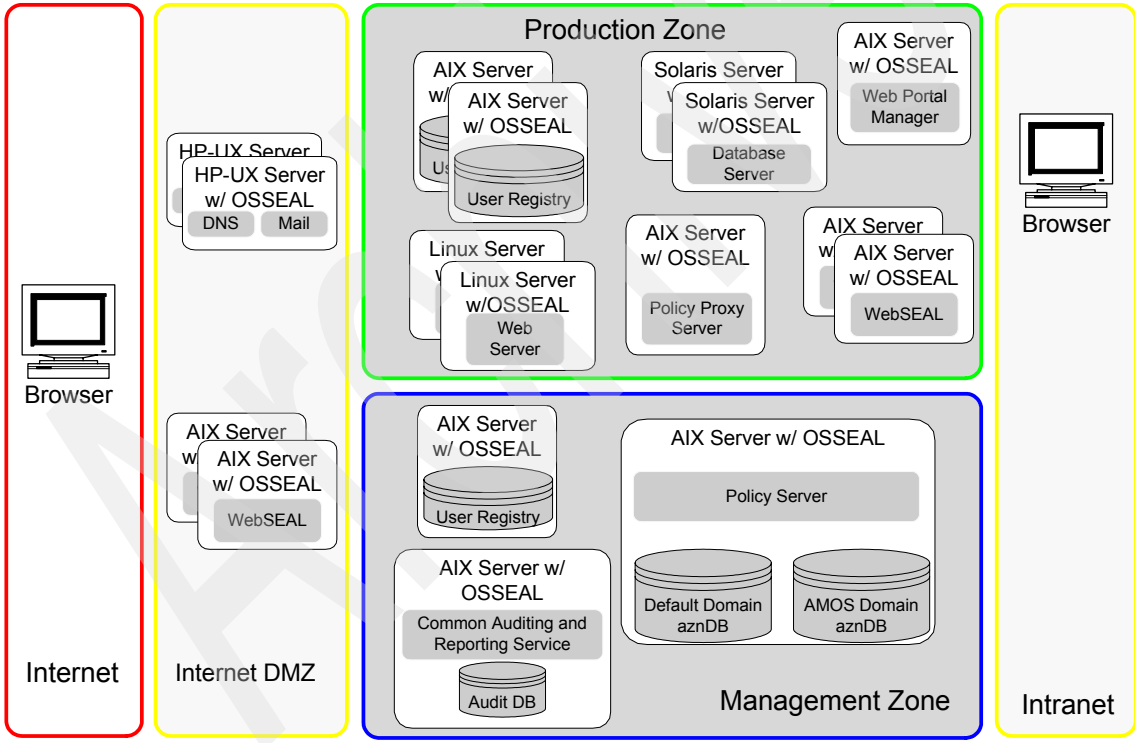


Figure 13-4 Stocks4u.com integrated security solution for UNIX and Web applications

The policy server at Stocks4u.com now has two distinct and separate databases to manage. This new database does not, however, require the addition of any new hardware.

13.6 Conclusion

In this chapter, we used the guidelines that we discussed previously in this book to illustrate the thought process involved in developing a typical Access Manager for Operating Systems solution architecture. With this example, you can further extrapolate to more complicated environments using the same thought process.

14

Access Manager for Business Integration

This chapter describes Tivoli Access Manager for Business Integration and the role that the product plays within the overall enterprise architecture. This is described in the contexts of the security architecture subsystems.

Tivoli Access Manager for Business Integration and Tivoli Access Manager for WebSphere Business Integration Broker are the two resource managers that Access Manager uses to provide secure messaging through WebSphere MQ. The two products provide authenticated, authorized, and secured transactions between applications.

It is assumed that the reader has a basic understanding and knowledge of the core Access Manager infrastructure that is described in Chapter 5, “Access Manager core components” on page 163.

14.1 Product overviews

The following sections briefly introduce IBM WebSphere MQ, WebSphere Business Integration Message Broker, Access Manager for Business Integration, and Access Manager for WebSphere Business Integration Brokers.

14.1.1 IBM WebSphere MQ

Message queuing (MQ) is a method of application-to-application communication. Applications communicate by writing and retrieving application-specific data (messages) to and from queues without having a private, dedicated connection to link them. Messaging means that programs communicate with each other by sending data in messages and not by calling each other directly, which is the case for technologies such as remote procedure calls. Queuing means that applications communicate through queues. The use of queues removes the requirement for both the sending and receiving applications to be executing concurrently.

IBM WebSphere MQ products enable applications to communicate with each other across a network of different components, such as processors, subsystems, operating systems, and communication protocols. For example, IBM WebSphere MQ supports more than 35 different operating systems.

IBM WebSphere MQ supports two different application programming interfaces: Java Message Service (JMS) and Message Queuing Interface (MQI). On IBM WebSphere MQ servers, the JMS binding mode is mapped to the MQI. An application talks directly to its local queue manager by using MQI, which is a set of calls that request services from the queue manager. The attractive feature of MQI is that it provides only 13 calls. This means that it is a very simple interface for application programmers to use, because most of the hard work is done transparently.

Figure 14-1 on page 427 shows the essence of IBM WebSphere MQ programming. The first step is for the application to connect to the queue manager. It does this through the MQConnect call. The next step is to open a queue for output using the MQOpen call. The application then puts its data on the queue using the MQPut call. To receive data, the application calls the MQOpen call to open an input queue. The application receives data from the queue using the MQGet call.

Also shown in the figure are the message channel agent (MCA), channel exits, and object authority manager (OAM). The MCA is the IBM WebSphere MQ program that moves the messages from the local transmission queue to the target queue manager using existing transport services, such as TCP/IP and SNA. These transport services are known as channels. The channel exits are

user-written libraries that can be entered from one of a defined number of places during channel operation. The OAM is the default authorization service (OS-specific) for command and object management. These three components are important for existing security solutions for IBM WebSphere MQ.

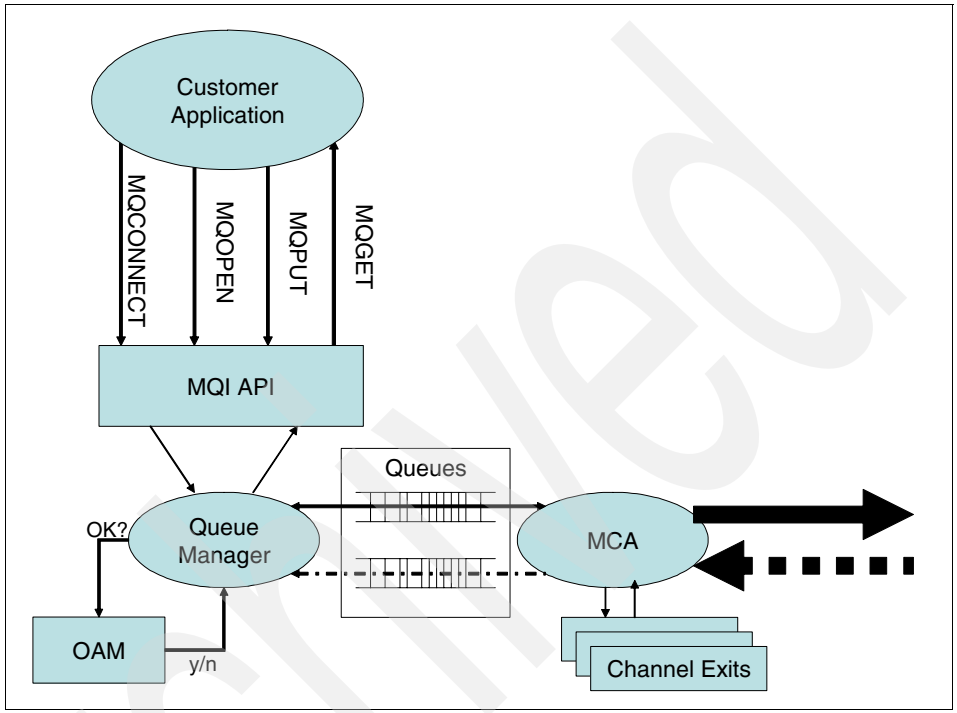


Figure 14-1 IBM WebSphere MQ programming

14.1.2 WebSphere Business Integration Message Broker

WebSphere Business Integration Message Broker enables information, packaged as messages, to flow between different business applications, ranging from large existing systems to unmanned devices such as sensors or pipelines.

WebSphere Business Integration Event Broker and Message Broker provide the capability to integrate resources without bounds by mediating between message transports and message formats and by routing messages on behalf of the enterprise. WebSphere Business Integration Event Broker is a true subset of WebSphere Business Integration Message Broker. In other words, a Message Broker is an Event Broker with additional capabilities.

WebSphere Business Integration Message Broker provides powerful publish/subscribe capability in a Java Messaging System (JMS) environment.

Component descriptions

Figure 14-2 shows the interaction between various components of WebSphere Business Integration Message Broker. A brief description of the components shown in the diagram follows.

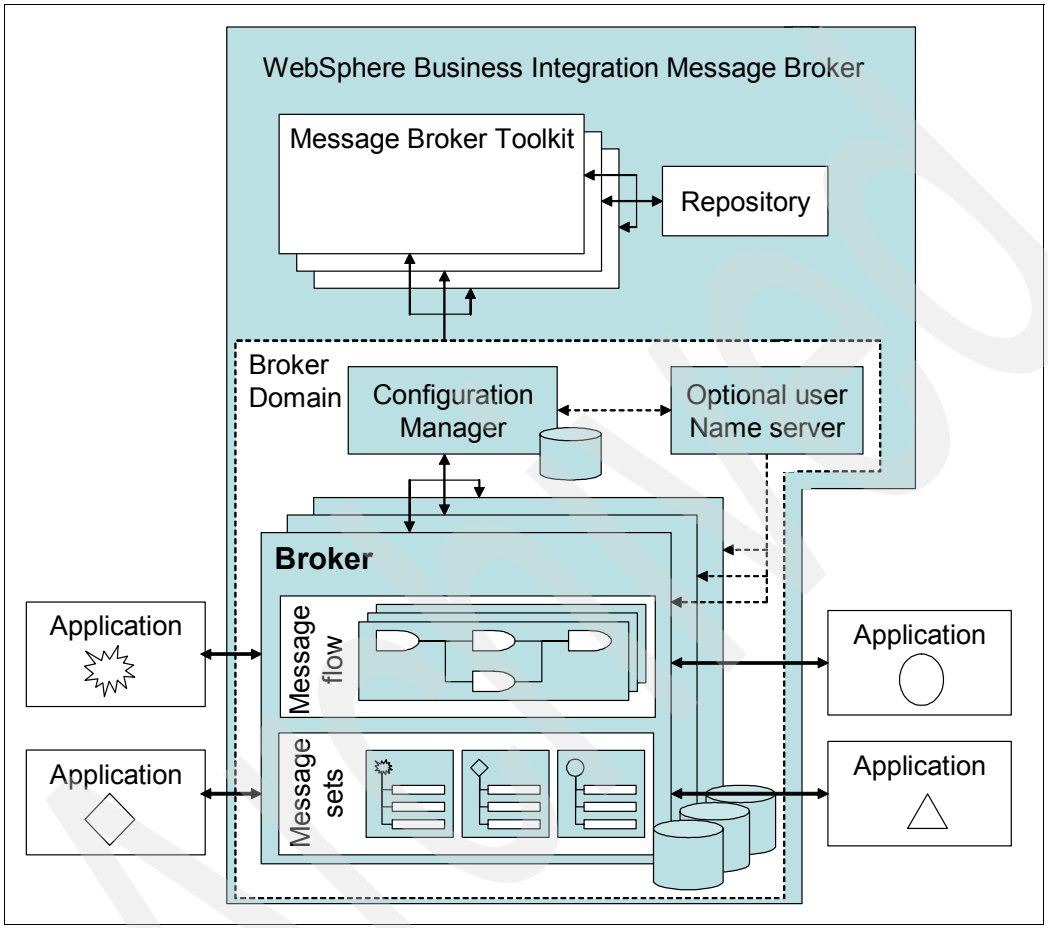


Figure 14-2 WebSphere Business Integration Message Broker overview

Broker

The broker is a system service on Windows platforms, or a server process on UNIX platforms, that controls processes that run message flows. Applications send messages to the broker using WebSphere MQ queues and connections. The broker routes each message using the rules defined in message flows and message sets, and transforms the data into the structure required by the receiving application.

The broker uses sender and receiver channels to communicate with the Configuration Manager and other brokers in the broker domain.

The broker depends on a broker database to hold broker information. This information includes control data for resources defined to the broker, such as deployed message flows. The database is also known as the broker's local persistent store.

The broker connects to the database using an ODBC connection.

When you create a broker, you must give it a name that is unique within the broker domain. Broker names are case-sensitive on all supported platforms except Windows platforms.

Broker domain

A broker domain is one or more brokers that share a common configuration, together with the single Configuration Manager that controls them.

You install, create, and start one or more brokers, and an optional User Name Server, in a broker domain. You can configure more than one broker domain, each managed by its own Configuration Manager.

User Name Server

The User Name Server is an optional run-time component that provides authentication of users and groups performing publish/subscribe operations. If you have applications that use the publish/subscribe services of a broker, you can apply an additional level of security to the topics on which messages are published and subscribed. This additional security, known as topic-based security, is managed by the User Name Server. It provides administrative control over who can publish and who can subscribe.

Configuration Manager

The Configuration Manager is the interface between the WebSphere Business Integration Message Brokers Toolkit in the configuration repository and an executing set of brokers. It provides brokers with their initial configuration and updates them with any subsequent changes. It maintains the broker domain configuration.

The Configuration Manager is the central run-time component that manages the components and resources that constitute the broker domain.

The Configuration Manager has four main functions:

- Maintains configuration details in the configuration repository. This set of database tables provides a central record of the broker domain components.
- Deploys the broker topology and message-processing operations in response to actions initiated through the Toolkit. Broker archive (bar) files are deployed through the Configuration Manager to the execution groups within a broker.
- Reports on the results of deployment and the status of the broker.
- Communicates with other components in the broker domain using WebSphere MQ transport services.

You must install, create, and start a Configuration Manager for each broker domain.

Message flow

A message flow is a directed graph of message flow nodes that represents the actions that are performed on a message when it is received and processed by a broker. Each node in a message flow represents a processing step, and the connections in the flow determine which processing steps are carried out, and in which order. A message flow must include an input node that provides the source of the messages that are processed. A message flow represents a set of actions that can be executed by a broker and therefore can be deployed.

Message sets

A message set is a container, a logical grouping of messages and associated message resources (elements, types, groups).

Execution group

An execution group is a named grouping of message flows that have been assigned to a broker. The broker enforces a degree of isolation between message flows in distinct execution groups by ensuring that they execute in separate address spaces, or as unique processes. Within an execution group, the assigned message flows run in different thread pools.

Each execution group is started as a separate operating system process, providing an isolated runtime environment for a set of deployed message flows. A single default execution group is set up ready and for use when you create a broker in the Toolkit.

Execution groups are created and deployed in the Toolkit. Tivoli Access Manager for WebSphere Business Integration Brokers supports one execution group per broker.

Term descriptions

We now describe some important terms and concepts that must be understood before using Tivoli Access Manager for WebSphere Business Integration Brokers.

Publish/subscribe

Publish/subscribe is a style of messaging application in which the applications that provide information are decoupled from the applications that might use that information. The following terms are used in a publish/subscribe system:

Publisher	An application that provides information
Subscriber	An application that uses the information
Publication	The information that a publisher provides
Subscription	The request that a subscriber makes for information

In a publish/subscribe system, a publisher does not need to know who uses the information that it provides, and a subscriber does not need to know who provides the information that it uses.

Message brokers make sure that messages arrive at the correct destinations and are transformed into the format required at each destination.

The simplest form of a publish/subscribe system has one message broker, one application that publishes messages, and one application that subscribes to messages, as shown in Figure 14-3.

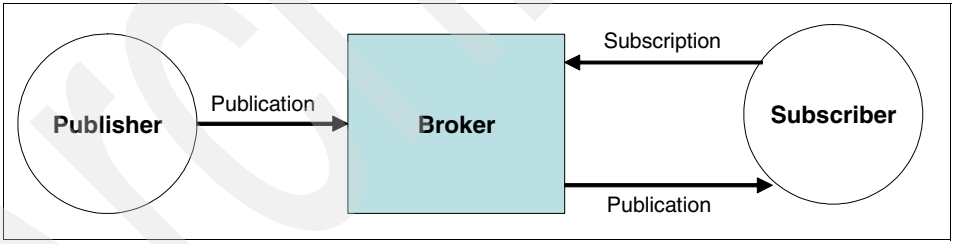


Figure 14-3 Simple publish/subscribe system

Topic

A topic is a character string that describes the nature of the data that is being published in a publish/subscribe system.

Topics are key to the successful delivery of messages in a publish/subscribe system. Instead of including a specific destination address in each message, a publisher assigns a topic to the message. The message broker matches the topic with a list of clients who have subscribed to that topic and delivers the message to each of those clients.

Publish/subscribe security

A secure publish/subscribe system needs at least these two security services:

- Topic-based security

  Access to messages on particular topics is controlled using access control lists (ACLs).

- Authentication services

  An authentication protocol is used by a broker and a client application to confirm that they are both valid participants in a session.

14.1.3 Access Manager for Business Integration

IBM Tivoli Access Manager for Business Integration operates in conjunction with the base components provided by IBM Tivoli Access Manager. Together, these software applications provide a security solution for IBM MQSeries, Version 5.2, and IBM WebSphere MQ, Version 5.3, products. All subsequent general references refer to IBM WebSphere MQ.

With IBM Tivoli Access Manager for Business Integration you can:

- Secure sensitive or high-value messages processed by IBM WebSphere MQ.
- Control which users have access to specific queues.
- Detect and remove rogue or unauthorized messages before they are processed by a receiving application.
- Generate detailed audit records showing which messages were expressly authorized and encrypted.
- Centrally define authorization and data protection policies for IBM WebSphere MQ resources (getting and putting messages to queues).
- Provide integrity and privacy protection for your data as it flows across the network and while it is in a queue.
- Secure existing off-the-shelf and customer-written applications for IBM WebSphere MQ.

IBM Tivoli Access Manager for Business Integration furnishes IBM WebSphere MQ applications with the following functionality:

- A centralized authorization service that defines security policies for IBM WebSphere MQ queues and messages in these queues.
- Privacy, in the form of encryption, and integrity in the form of checks against message modification, so that senders and receivers of IBM WebSphere MQ messages can exchange them with security. IBM Tivoli Access Manager for

Business Integration provides these services while the messages are in transit as well as when the messages are stored in the queues.

- IBM Tivoli Access Manager for Business Integration identifies IBM WebSphere MQ users with X.509 distinguished names that are independent of the operating system and network.
- Transparent message-level security. IBM WebSphere MQ applications do not have to be modified to be protected by IBM Tivoli Access Manager for Business Integration.

14.1.4 Access Manager for WebSphere Business Integration Brokers

IBM Tivoli Access Manager for WebSphere Business Integration Brokers, in conjunction with Access Manager, provides the security solution for WebSphere Business Integration Message Broker Version 5.0 and WebSphere Business Integration Event Broker, Version 5.0. All subsequent references refer to this product as Message Broker. With Tivoli Access Manager for WebSphere Business Integration Brokers you can:

- Define authorization policies centrally for Java Message Service (JMS) publish/subscribe topics.
- Secure JMS publish/subscribe applications using Tivoli Access Manager authentication.
- Provide user name/password or credential-based authentication for JMS publish/subscribe applications.
- Provide an audit trail for authorization events in WebSphere Business Integration Message Broker.

14.2 Architectural perspective

We use the concept of security architected subsystems, as discussed in 2.1, “Common security architecture subsystems” on page 20, to provide a way to group common attributes and to provide a common set of services to a broad range of applications. The Subsystem approach allows for a clear articulation and understanding of the security solution, and enables this to be deployed as a service within a real-world infrastructure.

The main subsystems addressed by Access Manager for Business Integration are:

- Access control: Access Manager is used to authenticate users and to enforce security policy at an application and system level.
- Auditing: The Access Manager components and infrastructure provide a comprehensive logging framework that can be integrated with most threat management system.
- Information flow control: Access Manager for Business Integration controls the flow of information over MQ between applications in terms of authorization and message protection through the use of cryptographic confidentiality and integrity mechanisms.

Access Manager for Business Integration utilizes all of the subsystems. However, the three just listed are fundamental to subsystems involving Access Manager within an overall Enterprise Architecture.

14.2.1 Design principles

The design of any architecture must be based on clearly defined and articulated principles that form a foundation for the design process. Whenever in doubt about a design decision, the principles should be used to map a path forward and to justify the overall design.

Some key principle can be applied to an access control solution:

- The security solution must have a central point of authority for security-related information. This authority must support both centralized and distributed management.
  - Motivation: This principle drives the need for one source of authoritative security-related policy within an organization. It enables a consistent policy to be applied across applications and systems and throughout the organization while providing a flexible administration framework that will fit into and enhance an organization's operation capabilities.
  - Implication: This principle implies a high degree of integration, broad coverage, and flexibility required from the products that are chosen to support it. Integration is one of the greatest challenges.
- The solution should support the end-to-end flow of transactions throughout an environment. That is, security should be applied throughout the system and not just at the front door. Mechanisms should exist that not only authorize

transactions but also protect transaction data from tampering and eavesdropping.

  - Motivation: Many of today's online application support high-valued transactions that require appropriate security controls end-to-end. Also, privacy laws now force organizations to provide better data protection.
  - Implication: Authorization and cryptographic message protection throughout a distributed environment creates numerous integration issues. A distributed environment will have many requirements that are hard to satisfy with just one product.

- Sufficient logging is required to capture all authentication and access control decision events and logs. The level of logging should be based on business and security requirements, hence the security solution should provide comprehensive and flexible logging coverage that enables it to be customized.
  - Motivation: Because no security solution is foolproof, it is essential to keep good records of the transactions performed by your security system. An easily manageable method of dealing with these records is essential.
  - Implications: Strong integration is required to provide logging across multiple systems. Mechanisms must be in place to collect, filter, analyze, and report on audit data.

These principles are not intended to be comprehensive, but to highlight some core objectives of the security solution.

Access Manager for Business Integration supports all of these principles. The Access Manager family of products, when integrated throughout an environment, provides a comprehensive access control capability. The breadth of the Access Manager solution along with its open architecture and interfaces means that it is a perfect solution to provide the majority of an enterprises's access control capabilities.

14.3 Access Manager for Business Integration

IBM Tivoli Access Manager for Business Integration enables IBM WebSphere MQ applications to send data with confidentiality and integrity by using keys associated with the sending and receiving applications. Application-level data protection enhances the SSL channel security that is part of IBM WebSphere MQ, Version 5.3. This additional level of data protection is critical for customers needing to establish a Health Insurance Portability Accountability Act (HIPAA) compliant implementation of IBM WebSphere MQ, or for any customer using IBM WebSphere MQ to process other types of sensitive data, such as high-value financial transactions or Human Resources (HR) data. The Tivoli Access

Manager authorization service provides access control to IBM WebSphere MQ-based services and restricts which users or processes can and cannot access messages on queues.

14.3.1 Security characteristics

IBM Tivoli Access Manager for Business Integration provides a high level of security in terms of user authorization and data protection while not affecting the end applications. IBM Tivoli Access Manager for Business Integration provides the following security benefits:

- Secures sensitive and high-value transactions processed by IBM WebSphere MQ.
- Controls which users and applications have access to specific queues.
- Detects and removes rogue or unauthorized messages before they are processed by a receiving application.
- Generates detailed auditing records.
- Verifies that messages were not modified while in transit from queue to queue.
- Centrally defines authorization policies (including data protection) for IBM WebSphere MQ resources (getting and putting messages to queues) using a common console for heterogeneous servers across the enterprise.
- Protects the data not only as it flows across the network but also as it sits in a queue.

Secures existing off-the-shelf and customer-written applications for IBM WebSphere MQ.

14.3.2 Architecture

Figure 14-4 shows a diagram of the core IBM Tivoli Access Manager for Business Integration components and security infrastructure components (in shaded areas).

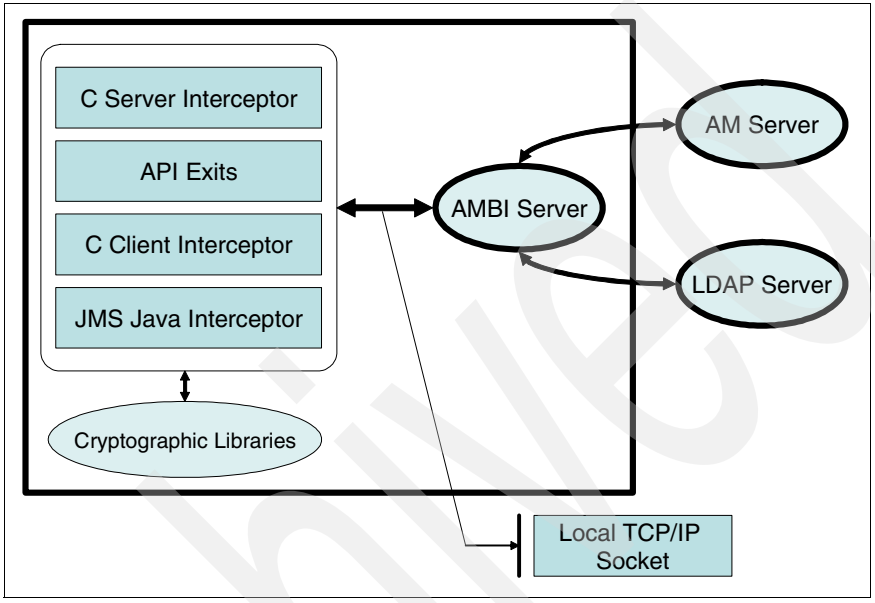


Figure 14-4 IBM Tivoli Access Manager for Business Integration environment

IBM Tivoli Access Manager for Business Integration relies on IBM Tivoli Access Manager servers and clients, and on the following Tivoli Access Manager services:

- Enterprise LDAP user registry where IBM WebSphere MQ users are represented as Tivoli Access Manager users.
- Centralized Policy Server to define authorization and data protection policy for access to IBM WebSphere MQ resources, such as queues.
- GUI-based management console provided by Tivoli Access Manager Web Portal Manager.

IBM Tivoli Access Manager for Business Integration uses the industry standard aznAPI to obtain authorization services from the Policy Server.

For a more detailed discussion of the core components refer to Chapter 5, “Access Manager core components” on page 163.

14.3.3 Components and dependencies

IBM Tivoli Access Manager for Business Integration supports two different interception environments:

- IBM Tivoli Access Manager for Business Integration Server and Client Interceptors.

  The key piece of IBM Tivoli Access Manager for Business Integration is a set of multi-threaded, shared libraries that executes in the process space of an IBM WebSphere MQ application. The IBM Tivoli Access Manager for Business Integration libraries intercept IBM WebSphere MQ C API calls and enable IBM WebSphere MQ applications to be secured without any changes.

- IBM Tivoli Access Manager for Business Integration JMS Interceptor.

  IBM Tivoli Access Manager for Business Integration Java Message Service (JMS) Interceptor is a set of JAR files and tools to intercept JMS calls and enable the JMS application to be secured without any changes.

Access Manager for Business Integration server

Access Manager for Business Integration server is a service on Windows and UNIX platforms. It accepts service requests from all IBM Tivoli Access Manager for Business Integration Interceptors running on the local system and provides the following services:

- Authorization checks based on the security policy specified using the IBM Tivoli Access Manager administrative tools.
- Auditing of security events, such as mapping of the PKI identity to the IBM Tivoli Access Manager user.
- Retrieval of security policy information, such as signature algorithms, encryption strength, queue resolution, and so on.
- Public key certificate retrieval for recipients of messages.

Access Manager C authorization APIs are used to perform authorization checks and obtain security policy information. They also communicate with the LDAP server to obtain user mapping information and retrieve public key certificates for recipients of messages. IBM Tivoli Access Manager for Business Integration server is also responsible for generating audit records for security events according to user-defined audit policy.

IBM Tivoli Access Manager for Business Integration server is optimized to serve a large number of WebSphere MQ applications. It is a *local-mode* IBM Tivoli Access Manager authorization API application that uses a local copy of the security policy database. The policy database is updated by periodic notifications from the IBM Tivoli Access Manager Policy Server. It can also be updated using the IBM Tivoli Access Manager administrative tools.

Access Manager for Business Integration Interceptor model

IBM Tivoli Access Manager for Business Integration operates as shown in Figure 14-5.

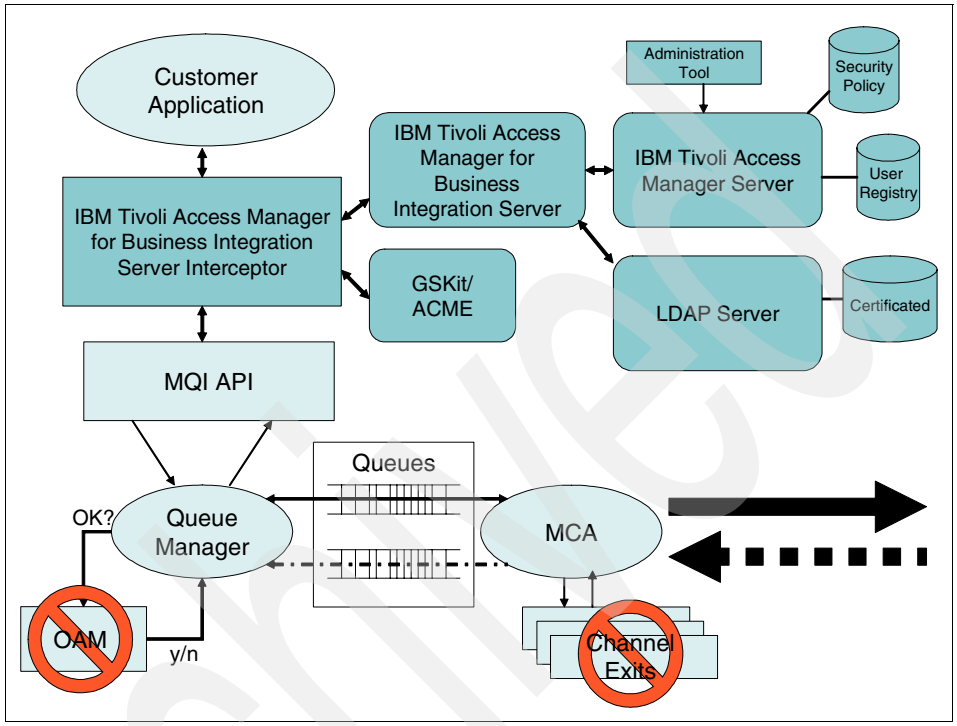


Figure 14-5 Tivoli Access Manager for Business Integration interceptor model

IBM Tivoli Access Manager for Business Integration provides a set of multi-threaded, shared libraries that executes in the process space of the IBM WebSphere MQ application. The IBM Tivoli Access Manager for Business Integration libraries intercept IBM WebSphere MQ calls, enabling IBM WebSphere MQ applications to be secured without any modifications.

On IBM WebSphere MQ Version 5.3 servers, IBM Tivoli Access Manager for Business Integration intercepts IBM WebSphere MQ calls by using the API exits mechanism. If the IBM Tivoli Access Manager for Business Integration API exit is added to the configuration of some or all queue managers, IBM WebSphere MQ will call into this library before and after every MQI call. For more information about API exits, see the *WebSphere MQ System Administration Guide*.

API exits are not supported on IBM WebSphere MQ Version 5.2 servers or any IBM WebSphere MQ clients. On these platforms, interception is accomplished by

replacing the native IBM WebSphere MQ with the IBM Tivoli Access Manager for Business Integration interceptor library.

When intercepting an IBM WebSphere MQI call, IBM Tivoli Access Manager for Business Integration determines:

- Whether the request for IBM WebSphere MQ services is authorized.
- Whether the data in the transaction should be digitally signed or digitally encrypted, or both, before being placed in the queue requested.
- Whether a message has been signed (that the signature associated with the message is verified before the original message is presented to the requesting application).
- Whether a message has been encrypted, that the message is decrypted, and that the original message is presented to the requesting application.

Authentication

Access Manager for Business Integration uses Public Key Infrastructure (PKI) credentials to authenticate the user or application requesting IBM WebSphere MQ services. It also uses cryptographic client services to securely wrap messages using the IETF CMS standard. This encapsulation protects the message data from being disclosed or tampered with while in a queue or in transit.

Authorization and permission bits

IBM WebSphere MQ queues must be represented in the protected object space so that access policies can be attached to them.

When IBM Tivoli Access Manager for Business Integration is configured, it creates an object space container, which contains entries for each of the protected queues in the domain. Each queue is listed underneath its queue manager.

The ACL on the queue is checked when the application attempts to open the queue using MQOPEN after connecting to the queue manager. The mode in which the queue is opened, either for input (MQPUT) or output (MQGET), determines which permission bits in the ACLS are required.

Data protection and audit

There are three main functions supported in this category:

Integrity If integrity is specified, then all messages put onto the queue must be signed so that tampering can be detected and so that the sender is known.

Privacy Each message put onto the queue must be encrypted so that only the intended recipients can read it.

Auditing Tivoli Access Manager for Business Integration records each transaction involving the queue in question. Auditing records and notifies any violation to the integrity and privacy functions.

Error handling

IBM Tivoli Access Manager for Business Integration defines an error handling queue to manage messages that contain errors or messages that cannot be routed correctly. For example, if the message is signed when it should be encrypted, or if encryption or signature verification fails, then the message is sent to the error-handling queue.

IBM WebSphere MQ Client overview

An IBM WebSphere MQ client is part of the IBM WebSphere MQ product that can be installed on its own, on a separate machine from the MQ server. You can run an IBM WebSphere MQ application on an IBM WebSphere MQ client, which interacts with one or more IBM WebSphere MQ servers. The IBM WebSphere MQ client connects to queue managers by means of a communications protocol. The servers to which the client connects might or might not be part of an IBM WebSphere MQ cluster.

The queues and other IBM WebSphere MQ objects are held on a queue manager that you have installed on an MQ server machine. When the application issues a client call, the IBM WebSphere MQ client directs the request to a queue manager, where it is processed and from where a reply is sent back to the IBM WebSphere MQ client. The link between the MQ application and the IBM WebSphere MQ client is established dynamically at runtime.

Client interceptor considerations

The C Client Interceptor supports IBM WebSphere MQ 5.2 and IBM WebSphere MQ 5.3. The C Client Interceptor operates as shown in Figure 14-6.

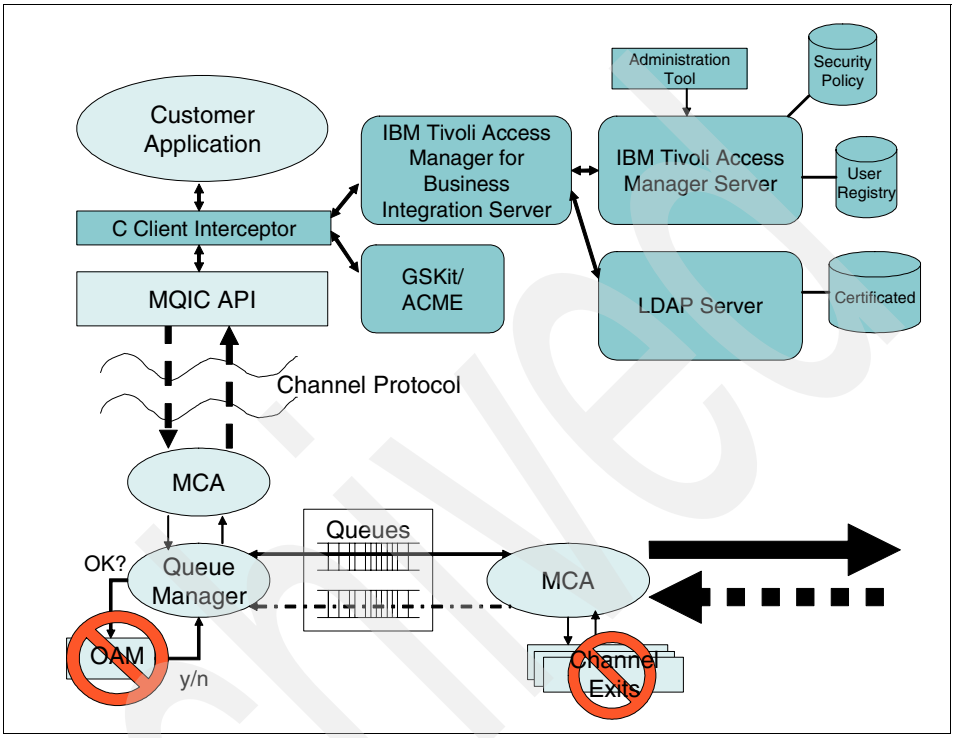


Figure 14-6 IBM Tivoli Access Manager for Business Integration C Client Interceptor

Currently, IBM WebSphere MQ does not permit user-written data conversion exits on the client. If message protection is in effect (integrity or privacy levels) and a message contains data in a user-defined format, the message is not converted, even if a data conversion exit for that format is defined on the IBM WebSphere MQ server. For security reasons, the C Client Interceptor cannot send plain text data back to the server for conversion. However, the C Client Interceptor converts messages in the IBM WebSphere MQ–defined message formats, such as Rules and Formatting Header (RFH).

JMS Interceptor considerations

IBM Tivoli Access Manager for Business Integration supports the IBM Tivoli Access Manager for Business Integration Java Message Service (JMS) Interceptor, which interoperates with other interceptors. The JMS Interceptor provides authorization, data protection, and auditing security services for the

JMS interfaces that are available as part of the IBM WebSphere MQ client support.

Note: Only applications that use the JNDI namespace are supported.

JMS Interceptor model

Figure 14-7 shows the architecture of a Java application that uses the JMS interfaces with an IBM WebSphere MQ JMS provider and the JMS Interceptor enabled. The JMS Interceptor is enabled using the pdmqjmsadmin program, which sets up the administered objects to use the JMS Interceptor. When the methods in these objects are invoked, authorization, audit, and data protection services are applied as per the security policy specified in the IBM Tivoli Access Manager protected object space.

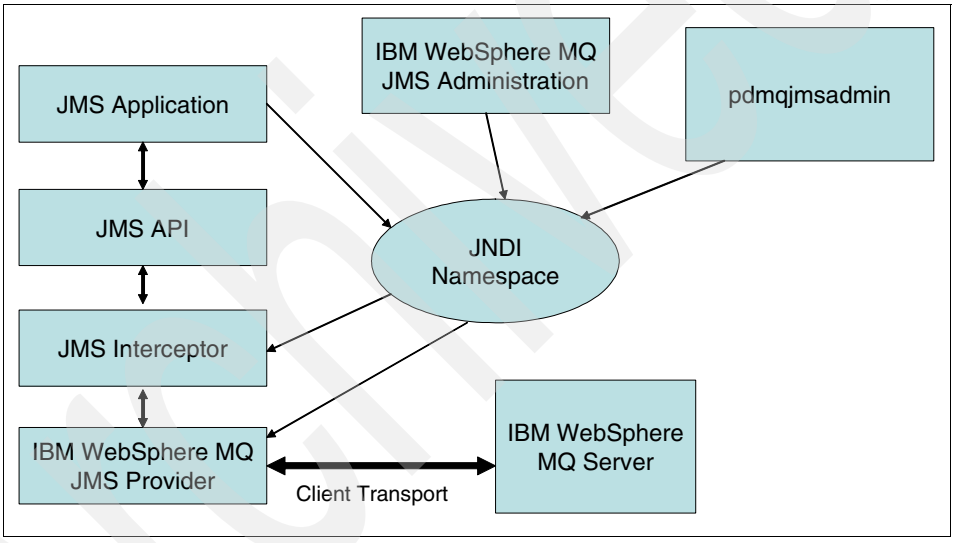


Figure 14-7 IBM WebSphere MQ JMS architecture with JMS Client Interceptor

The JMS Interceptor performs the same security services (authorization, data protection, and auditing) as the other interceptors provided by IBM Tivoli Access Manager for Business Integration.

14.4 Access Manager for WebSphere Business Integration Brokers

IBM Tivoli Access Manager for WebSphere Business Integration Brokers operates in conjunction with IBM Tivoli Access Manager, which is the base

component. Together, these software applications provide the security solution for WebSphere Business Integration Message Broker Version 5.0 and WebSphere Business Integration Event Broker Version 5.0. All subsequent references refer to this product as Message Broker. With Tivoli Access Manager for WebSphere Business Integration Brokers you can:

- Define authorization policies centrally for Java Message Service (JMS) publish/subscribe topics.
- Secure JMS publish/subscribe applications using Tivoli Access Manager authentication.
- Provide user name/password or credential-based authentication for JMS publish/subscribe applications.
- Provide an audit trail for authorization events in WebSphere Business Integration Message Broker.

Tivoli Access Manager for WebSphere Business Integration Brokers provides authorization services to the brokers. It also replaces the User Name Server to provide support for client authentication. Tivoli Access Manager for WebSphere Business Integration Brokers uses the centralized authorization policy support provided by Tivoli Access Manager to provide access control for protected resources or topics.

14.4.1 Authorization and permission bits

The publish and subscribe topics must be represented in the Tivoli Access Manager protected object space so that access policies can be attached to them.

When Tivoli Access Manager for WebSphere Business Integration Brokers is configured, it creates an object space container that contains entries for each of the protected topics in the domain. The protected object space that is created follows the previously described topic tree model. Because attached policies are inherited down the object space, a policy attached to a parent topic affects all topics below it unless another policy is attached to a specific topic.

Tivoli Access Manager for WebSphere Business Integration Broker's action bits are used within the Tivoli Access Manager ACLs that are attached to the protected objects. These permission bits are used to determine whether a user can publish or subscribe on a given topic. The ACL on the topic is checked when the application attempts to publish or subscribe on the topic. If the user does not have the required permissions, an authorization failure error is sent back to the application.

The ACL for a topic uses the following permissions:

P The user is allowed to publish to the topic.

S The user is allowed to subscribe to the topic.
R The user is allowed persistence on the topic.

When an application attempts to publish or subscribe to a topic, Tivoli Access Manager for WebSphere Business Integration Brokers only checks that the application had permission to publish or subscribe to the topic. If the user does not have the required permission, the publish or subscribe call fails, and the user is not allowed to publish or subscribe to the topic.

14.5 A distributed application at Stocks-4u.com

We now look briefly at the use of Access Manager in a distributed situation, with IBM MQSeries as a solution component.

Expanding on our current scenario, Stocks-4u.com intends to deploy a new stock transaction record application that uses MQSeries for data exchange between front-end and back-end application components. In this case, the front-end application component is deployed in the San Diego IT Center, and the back-end component is deployed in Savannah on a corporate mainframe host. The front-end application component is embedded within a servlet running on an IBM WebSphere Application Server platform. Clients may access the application via WebSEAL. Figure 14-8 illustrates these components.

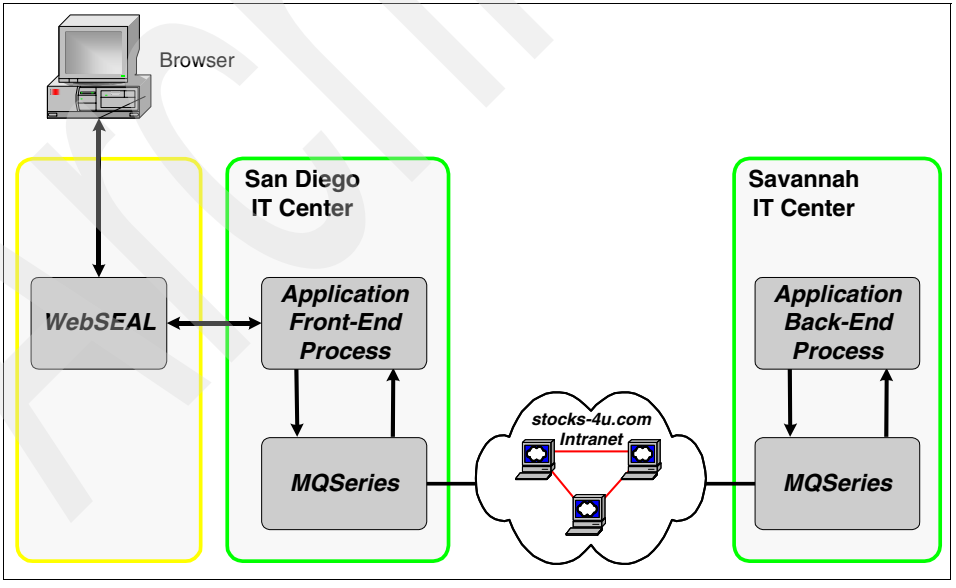


Figure 14-8 A Stocks-4u.com MQSeries application

In this case, the MQSeries channels represent a cross-site communication component that, depending on the specific network configuration, might not be secure. In this case, we assume that the MQSeries communication occurs across the Stocks-4u.com intranet, leaving the traffic largely unsecured. The question is: How can we use Access Manager to secure this communication and assure data privacy and integrity?

Access Manager for Business Integration is specifically designed to address such situations. As mentioned earlier, it provides queue security and transparently applies encryption to message channel traffic, permitting highly secure use of MQSeries over otherwise insecure channels.

Access Manager for Business Integration provides two key functions:

- It provides access control for enqueue and dequeue (put/get) operations using Access Manager's authorization engine.
- It can encrypt individual messages to protect their integrity and privacy. It does this transparent to the application.

Refer to Figure 14-9, which depicts Access Manager for Business Integration components and interactions.

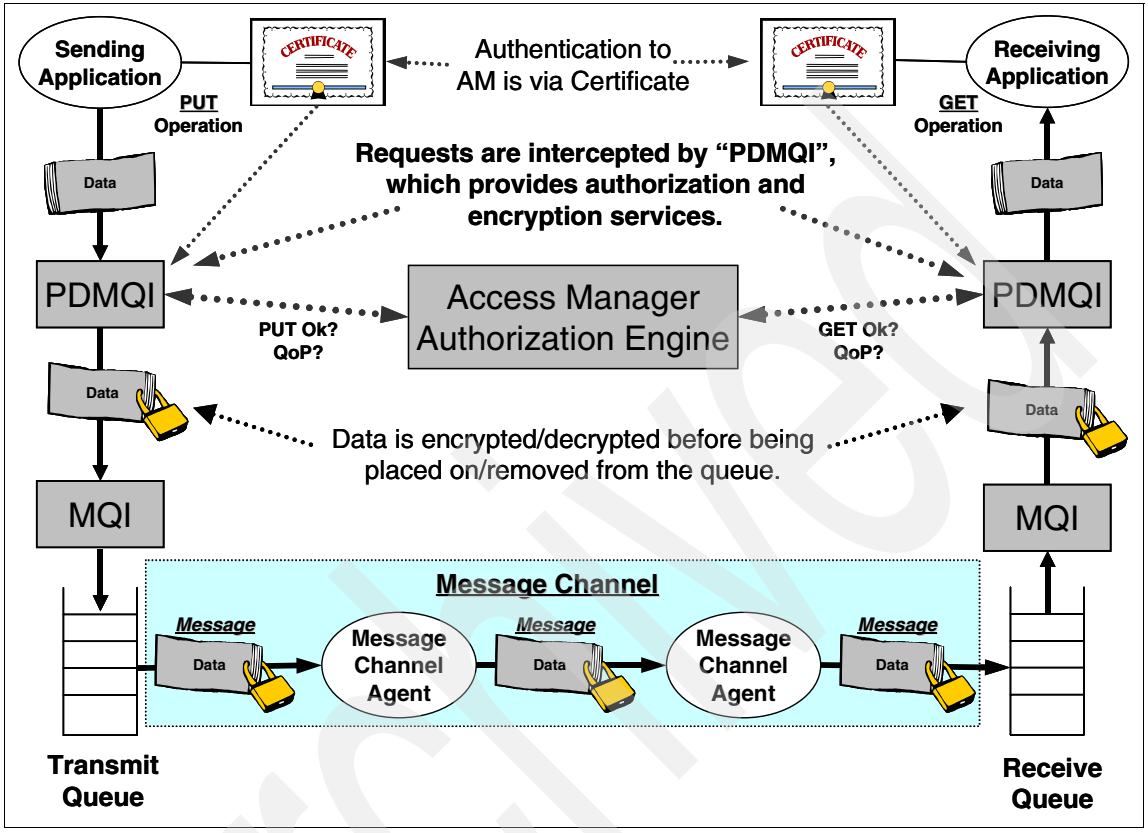


Figure 14-9 Access Manager for Business Integration architecture

Access Manager for Business Integration places a runtime library *shim* in the path between the application and MQSeries. The application continues to call the standard MQSeries runtime functions. These calls are intercepted transparently, where authorization checks and message encryption and decryption is done. Neither MQSeries itself nor the application are aware of Access Manager for Business Integration functions. Architecturally, this permits Access Manager for Business Integration to be deployed in existing MQSeries environments with minimal changes.

In the Stocks-4u.com environment, Access Manager for Business Integration can be *overlaid* on top of the MQSeries components used by the stock transaction application. Transaction records queued as MQ messages in San Diego are encrypted while in transit.

Finally, because queue access can now be managed, it is easier to leverage a single queue for multiple applications securely. Stocks-4u.com can deploy a common set of messaging channels used by all of its MQSeries applications, as shown in Figure 14-10.

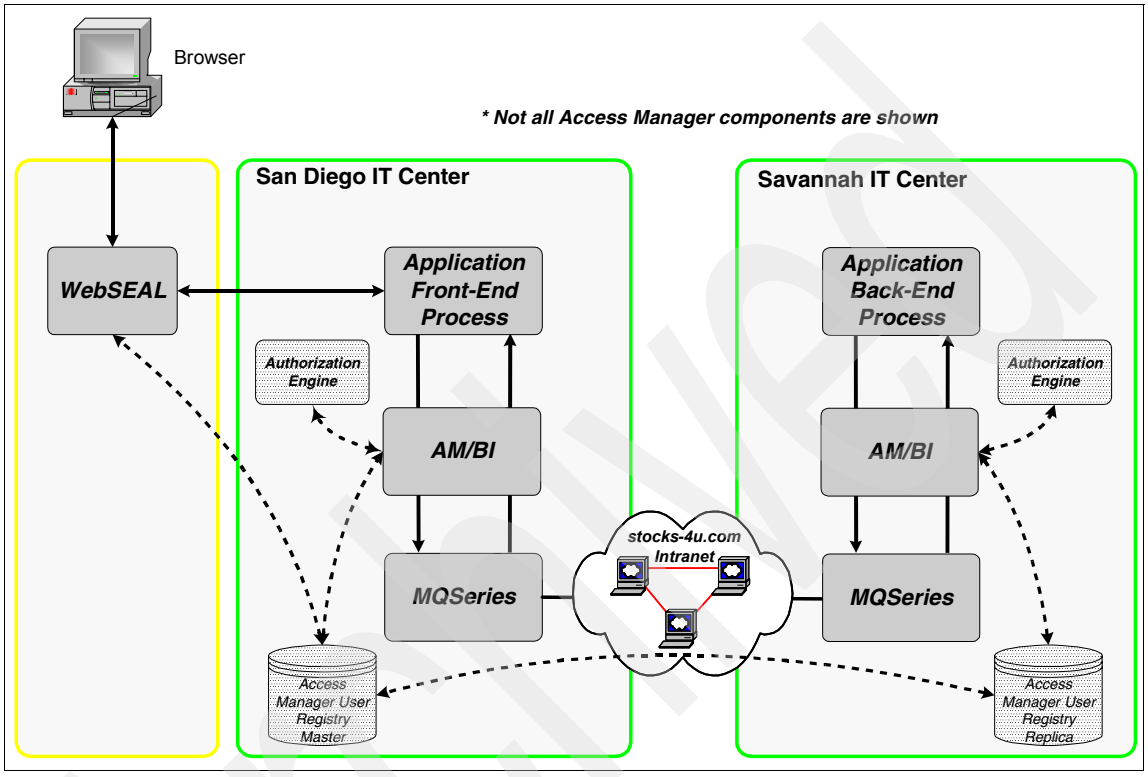


Figure 14-10 A Stocks-4u.com Access Manager for Business Integration scenario

Combining this with a WebSEAL front end, which is interfaced to the application front-end process, shows how Access Manager components may be utilized at multiple levels within the application framework to meet various security requirements.

15

Access Manager for Enterprise Single Sign-On

Whereas Tivoli Access Manager WebSEAL provides single sign-on for Web applications by inserting a reverse Web proxy in the network in front of all enterprise Web applications, Tivoli Access Manager for Enterprise Single Sign-On brings single sign-on to the desktop by placing an agent on the desktop to automatically respond to authentication requests on behalf of the user.

Tivoli Access Manager for Enterprise Single Sign-On detects and responds to all password-related events, automating every password management task for the user, including login, password selection, password change, and password reset. It delivers single sign-on for Windows, Web, Java, UNIX, Telnet, in-house developed, and host-based mainframe applications.

In this chapter we discuss the logical and physical architecture of Tivoli Access Manager for Enterprise Single Sign-On and its most fundamental components.

First we review the logical components within Tivoli Access Manager for Enterprise Single Sign-On and how they relate to each other, what is needed and what is an optional component.

Next on the physical architecture section we discuss how to physically deploy the components that make up Tivoli Access Manager for Enterprise Single Sign-On.

15.1 Logical component architecture

Tivoli Access Manager for Enterprise Single Sign-On provides single sign-on by introducing a secure middle layer that authenticates the user once and then automatically detects and handles subsequent requests for user credentials. Specifically, it uses patented client-side intelligence to respond to requests for user credentials (user name and ID, password, and so on) from any Windows, Web, or mainframe or host application. Tivoli Access Manager for Enterprise Single Sign-On supports authentication from any authenticator (for example, Passwords, Biometrics, Tokens/Smart Cards) and authentication service (for example, Windows, Entrust PKI, RSA Keon PKI, or LDAP directory).

Tivoli Access Manager for Enterprise Single Sign-On stores user credentials in an encrypted database using almost any encryption algorithm, including Triple-DES, AES/IES, RC4, Cobra, and Blowfish. Users can access their credentials from any workstation through Credential Synchronization (for example, Tivoli Directory Server and other LDAP directories, and Active Directory). Tivoli Access Manager for Enterprise Single Sign-On can log notifications of events (for example, logins) to almost any destination, such as SNMP and Windows Event Logging service.

Tivoli Access Manager for Enterprise Single Sign-On is designed to adapt to the specific needs of your organization. Many components provide several alternatives, allowing you to tailor your single sign-on deployment to fit your environment. Tivoli Access Manager for Enterprise Single Sign-On supports the most widely used authenticators, directory services, and PKIs, and can be customized through standard APIs to support less-common technologies.

Tivoli Access Manager for Enterprise Single Sign-On consists of the following logical components:

- Authentication
- Encryption
- Intelligent agent response
- Core (including storage)
- Credential synchronization
- Event logging
- Additional components

Figure 15-1 on page 451 illustrates these components.

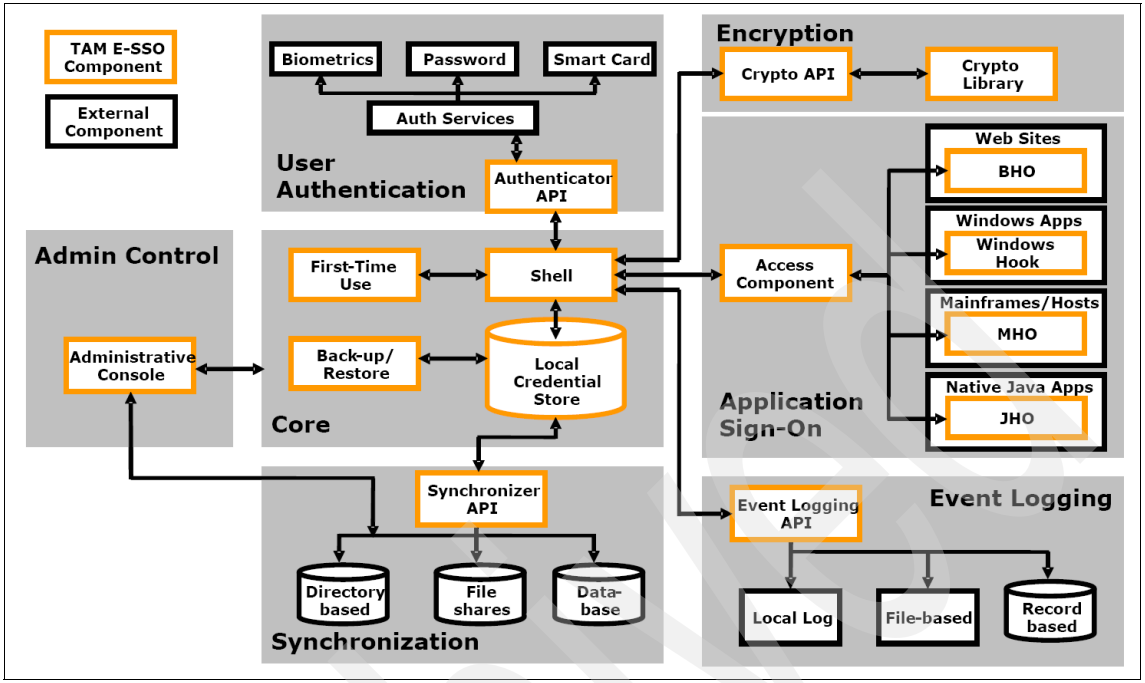


Figure 15-1 Logical architecture overview

Tivoli Access Manager for Enterprise Single Sign-On can be extended with the following adapters:

- Desktop Password Reset Adapter
- Authentication Adapter
- Provisioning Adapter
- Kiosk Adapter

In addition, administration is facilitated by the Administrative Console.

In the following sections we discuss every logical component in more detail.

15.1.1 Authentication

Authentication is how the system validates users so they gain access to Tivoli Access Manager for Enterprise Single Sign-On, for example, password, biometrics, token, and so on. The authentication component, depicted in Figure 15-2 on page 452, consists of the following three layers:

1. The Authenticator
2. System authentication services

3. Authenticator API

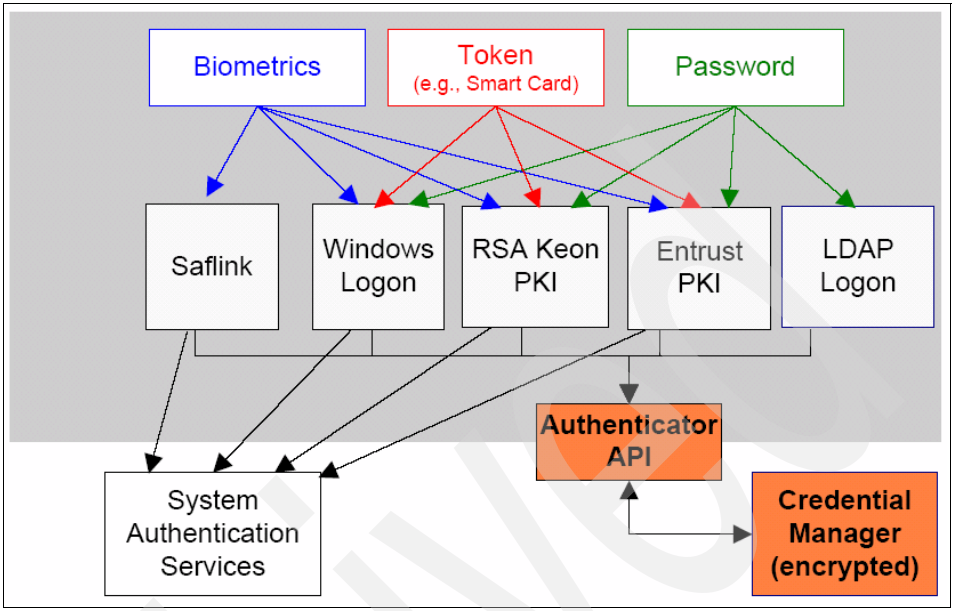


Figure 15-2 Authentication process

Authentication to Tivoli Access Manager for Enterprise Single Sign-On involves the following three steps:

1. The user provides credentials to the *authenticator*.
2. The authenticator validates the user with the *authentication service*.
3. The authentication service passes to the *authenticator API* information confirming validation and unlocking the user's encryption keys. See section 15.1.2, "Encryption" on page 453.

Other authenticators can be added and supported using the *Authentication Adapter*. The Authentication Adapter can add support for SmartCards (for example, Gemplus, and Schlumberger), signature authentication, iris recognition, tokens (for example, SAFLINK, Entrust Entelligence, RSA Keon, and NEC Touch Pass), digital client and server certificates, magnetic access cards, fingerprint biometrics (including Digital Persona biometrics), facial biometrics, handprint biometrics, voice print biometrics, and proximity cards such as Ensure's Xylock. The Authentication Adapter supports any form of strong authentication that replaces the Windows GINA (Graphical Identification and Authentication dynamic-link library), such as the RSA SecurID for Windows card. The RSA SoftID is supported as an application logon (not at the initial Windows authentication). Using the Tivoli Access Manager for Enterprise Single Sign-On

authentication API virtually any means of authentication can be supported by writing a specific authenticator for that product. More details on the Authentication Adapter are available in 15.1.9, “Authentication Adapter” on page 469.

Authenticator

An authenticator allows users to prove their identity, whether through a password, biometrics, or token. The authenticator takes the user’s proof and passes it to the authentication service. Tivoli Access Manager for Enterprise Single Sign-On ships with a set of authenticators, including Windows authentication, SmartCard, LDAP, RSA, and Entrust.

Authentication Service

The authentication service validates the credentials provided by the authenticator against either its own store or a system authentication service such as a Windows domain or a PKI. If validated, it passes the validation to the authenticator API. An authentication service can support a *disconnected* mode if it meets the requirements of the Tivoli Access Manager for Enterprise Single Sign-On authenticator API. This allows users to access their credentials even when the system authentication services are not available.

Note: Tivoli Access Manager for Enterprise Single Sign-On ships with six authentication services: Windows (Domain) Smart Card Logon (pass phrase and certificate based), LDAP, Entrust PKI, and enhanced versions of the Windows and LDAP authenticators that support pass phrase challenge.

Authenticator API

The Tivoli Access Manager for Enterprise Single Sign-On authenticator API is a set of plug-in interfaces that integrate the authentication user interface with the main Tivoli Access Manager for Enterprise Single Sign-On agent. It serves as a conduit between the authentication service and Tivoli Access Manager for Enterprise Single Sign-On. Third-party authentication services can integrate with Tivoli Access Manager for Enterprise Single Sign-On by utilizing the authenticator API.

15.1.2 Encryption

Encryption secures user credentials by creating a unique primary symmetric key for each user to be used in encrypting the user's credentials. End-to-end encryption is provided between the Tivoli Access Manager for Enterprise Single Sign-On agent and the directory using the selected encryption algorithm. The

Tivoli Access Manager for Enterprise Single Sign-On *default encryption algorithm* is the MS CAPI-provided *TripleDES*.

Credentials are stored encrypted on the client/PC, in transit, in memory, and in the directory. The only time that sensitive data is not encrypted is the moment a specific credential is requested for viewing (if permitted), or when it is submitted to an application for sign-on. The core requests that credentials be encrypted or decrypted based on the appropriate Crypto Library algorithm. The agent migrates credentials automatically to a new algorithm or strength (for example, from Triple DES to AES), if necessary. Tivoli Access Manager for Enterprise Single Sign-On supports a variety of encryption algorithms and algorithm strengths to suit most corporate, legal, security, performance, and other requirements. Figure 15-3 shows the encryption component.

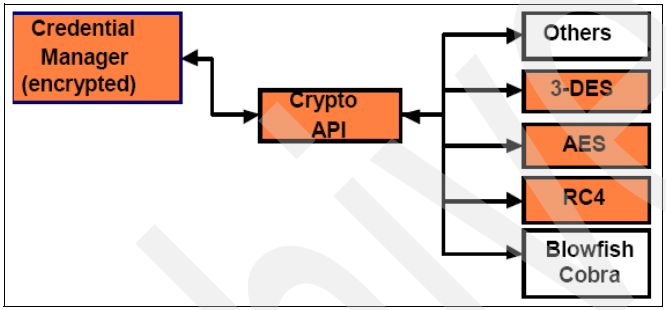


Figure 15-3 Encryption component

The advanced security setting controls the preferred encryption provider and strength. Tivoli Access Manager for Enterprise Single Sign-On ships with the following algorithms:

- Cobra 128-bit
- Cobra 128-bit (also)
- Blowfish 448-bit
- Triple DES 168-bit
- AES 256-bit
- Triple DES (MS-CAPI) (All OSs) default
- Triple DES (MS-CAPI) (XP/2003 only)
- RC-4 (MS-CAPI) (All OSs)
- RC-4 (MS-CAPI) (XP/2003 only)
- AES (MS-CAPI) (XP/2003 only)

To configure the *Default encryption*, go to your **Administrative Console** → **Global Agent Settings** → **Live** → **Security** → **Advanced**, as shown in Figure 15-4.

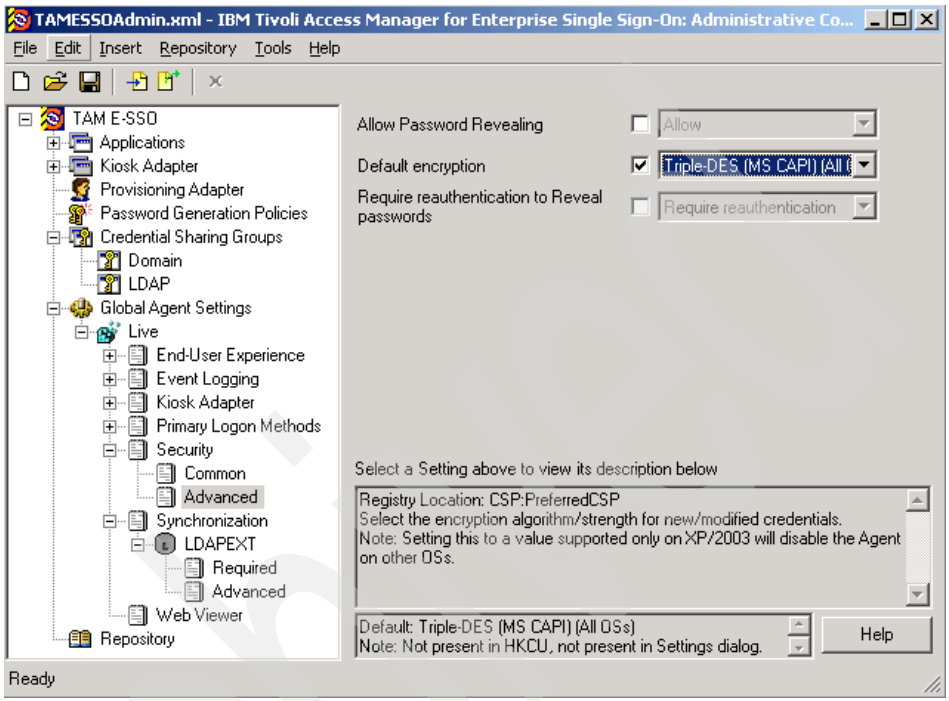


Figure 15-4 Default encryption setting control

Tivoli Access Manager for Enterprise Single Sign-On uses cryptography to confirm user authentication and to secure storage of user credential data. Upon first-time use, Tivoli Access Manager for Enterprise Single Sign-On generates and maintains a cryptographically unique *primary authentication key* that is authenticator independent and requires successful completion of the authentication process in order to be usable. Upon successful authentication, this key becomes available internally to Tivoli Access Manager for Enterprise Single Sign-On and is then used to decrypt and access user credentials. Each credential is only decrypted on an as-needed basis and is never stored or cached in the clear.

15.1.3 Intelligent agent response

When an application presents a request for credentials the agent detects this event, determines the appropriate action, and responds with the correct credentials. Tivoli Access Manager for Enterprise Single Sign-On ships with the

configuration information for many popular applications. Figure 15-5 depicts the logical architecture for the intelligent agent response process.

Configurations for common network/Web pop-up logins and for online service logins are stored in the configuration file applist.ini, which is located in the installation directory. For a more complete list of pre-configured applications, see **Reference** → **Preconfigured Applications** in the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console Help File.[1]

Note: All modules are installed by default but Mainframe/HOST and SAP support are disabled.

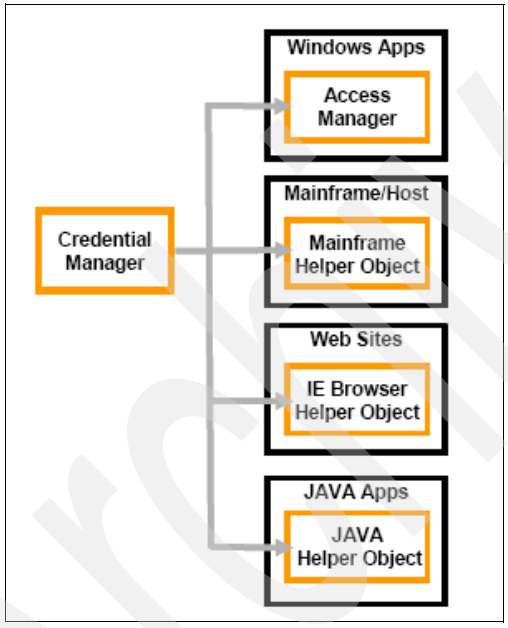


Figure 15-5 Intelligent client response process

Event detection

Tivoli Access Manager for Enterprise Single Sign-On detects requests for credentials in a variety of ways, depending on application type (Web, Windows, and Mainframe/Host).

[1] This help file is installed automatically with the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console.

Determine action

Tivoli Access Manager for Enterprise Single Sign-On determines whether the event is a logon or password change:

- If the dialog is a password-change dialog, Tivoli Access Manager for Enterprise Single Sign-On can be configured to either generate a new password and respond on behalf of the user or prompt the user to enter a new password.
- In case of a regular logon dialog Tivoli Access Manager for Enterprise Single Sign-On determines whether it has all necessary information, or if it needs to request information from the user.

  If user credentials are not present, Tivoli Access Manager for Enterprise Single Sign-On prompts the user for credentials. If the user provides credentials to Tivoli Access Manager for Enterprise Single Sign-On, the Shell stores the credentials in the *Local Credential Store* for future use.

Intelligent response

The Tivoli Access Manager for Enterprise Single Sign-On Access Component retrieves the credentials from the Local Credential Store and submits them to the application in the most effective and secure way possible for that application. For a password change, Tivoli Access Manager for Enterprise Single Sign-On then submits (in the most secure way possible for that application) the old password (if required), new password, and new password again for verification (if required). If the password change is not successful, the user can instruct Tivoli Access Manager for Enterprise Single Sign-On to use the old password.

In the following sections we take a closer look at the different kinds of applications that Tivoli Access Manager for Enterprise Single Sign-On supports:

- Windows applications
- Mainframe and host applications
- Web applications
- Java applications and applets

Windows applications

Tivoli Access Manager for Enterprise Single Sign-On responds to any and all requests for user credentials from Windows applications. It works without any special configuration after you install it with the most widely used applications. In addition, you can configure it to work with any other individual application.

All credential requests in Windows have specific attributes: application name, window name, the control ID of the input field, and so on. Tivoli Access Manager for Enterprise Single Sign-On looks for the specific attributes of each application's logon and password change dialog boxes and responds to these accordingly. The attributes are stored in the basic applist.ini and administrative

entlist.ini configuration files. See the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console help system for additional information.

The Tivoli Access Manager for Enterprise Single Sign-On hook (vgohook.dll) component captures standard OS-level Windows messages and sends them to the Tivoli Access Manager for Enterprise Single Sign-On Shell and Access Manager components of the Windows applications. When a specified application creates a dialog, Tivoli Access Manager for Enterprise Single Sign-On looks at the window title. If Tivoli Access Manager for Enterprise Single Sign-On recognizes the window title, it searches for the appropriate control ID or IDs.

Tivoli Access Manager for Enterprise Single Sign-On submits credentials to most Windows applications through secure, standard, OS-level Windows messages. Thus, keyboard-sniffing utilities cannot intercept the credentials. Furthermore, because Tivoli Access Manager for Enterprise Single Sign-On does not use scripts or keystrokes, users cannot confuse the response by selecting and working in another application.

Mainframe and host applications

Tivoli Access Manager for Enterprise Single Sign-On responds to any and all requests for user credentials from mainframe and host applications. It works without modification with the most popular mainframe and host emulators. In addition, you can configure it to work with others.

All requests for credentials in mainframe and host applications have specific attributes: window title and various blocks of text (at specific coordinates for Mainframe applications), user name and password field text, and so on. Tivoli Access Manager for Enterprise Single Sign-On looks for the specific attributes of each application's logon and password-change screens and responds accordingly. The attributes are stored in the administrative entlist.ini configuration file. See the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console help system for additional information.

The Tivoli Access Manager for Enterprise Single Sign-On *Mainframe Helper Object* monitors emulators, looking for the defined matches. When a new panel is presented, Tivoli Access Manager for Enterprise Single Sign-On reviews the text for matching fields. If all strings match, Tivoli Access Manager for Enterprise Single Sign-On uses the Mainframe Helper Object to submit user credentials.

Tivoli Access Manager for Enterprise Single Sign-On submits credentials to most emulators through HLLAPI. Thus, keyboard-sniffing utilities cannot intercept these credentials. Furthermore, because Tivoli Access Manager for Enterprise Single Sign-On does not use scripts or keystrokes for these emulators, users cannot confuse the response by selecting and working in another application.

Note: Tivoli Access Manager for Enterprise Single Sign-On also supports some emulators that have a scripting language capable of presenting a (hidden) dialog box to the user.

Web applications

Tivoli Access Manager for Enterprise Single Sign-On responds to any and all requests for user credentials from Web applications, whether in a form or through a pop-up dialog. Unlike most single sign-on products, Tivoli Access Manager for Enterprise Single Sign-On supports access to all Web applications, not just intranet applications. Most Web applications are supported without modification and new applications can be added.

All credential requests in Web applications are either in *forms* or in *dialog boxes*. The Tivoli Access Manager for Enterprise Single Sign-On *Browser Helper Object* (BHO) and *Event Manager* respond to the specific events of a Web dialog box opening or of a Web page rendering.

There is one BHO for Citrix/Terminal Services environments and another for standard Windows environments. Both BHOs detect events from the browser and can directly interact within the browser engine. The standard BHO also supports Internet Explorer embedded within Lotus Notes. Because Tivoli Access Manager for Enterprise Single Sign-On does not use scripts or keystrokes for Internet Explorer, users cannot confuse the response by selecting and working in another application.

Tivoli Access Manager for Enterprise Single Sign-On handles the two types of Web application credential requests similarly, as follows:

- Pop-up dialog boxes

  Pop-up dialog boxes have specific attributes: realm, site, and so on. Tivoli Access Manager for Enterprise Single Sign-On understands the specific attributes of each application's logon and password-change screens and responds accordingly. The attributes are stored in the basic applist.ini and administrative entlist.ini files. When a new pop-up dialog box is created, Tivoli Access Manager for Enterprise Single Sign-On reviews the dialog, requests credentials from the shell, and then submits them to the pop-up dialog box.

- Forms

  Forms have specific attributes: URL (including domain), frame name, form name, specific blocks of text on the page, user name and password field text, password fields (`HTML <Input type=password>`), and so on. Tivoli Access Manager for Enterprise Single Sign-On looks for the specific attributes of each application's logon and password-change screens and responds accordingly. The attributes are stored in the basic applist.ini and

administrative entlist.ini files. When a new page is fully rendered, the BHO reviews the page for matching criteria. If at least a password field is present, the BHO requests credentials from the shell then injects them into the browser.

Java applications and applets

Tivoli Access Manager for Enterprise Single Sign-On responds to login and password change requests for virtually all AWT (Abstract Window Toolkit) and Swing Java applications and applets built on the Sun Java Runtime Engine 1.4.1 or higher. New Java applications or applets can be supported by using the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console.

15.1.4 Core (including storage)

Tivoli Access Manager for Enterprise Single Sign-On stores user credentials locally in the encrypted *Local Credential Store*. It never maintains credentials unencrypted on disk or in memory. See 15.1.2, “Encryption” on page 453. The credentials are locally stored and encrypted for each user in a secure database file. Within this file are the encrypted records for each set of user credentials, user settings, and additional configuration information. Because the credential file is stored locally, there is no risk that a disruption in a centralized server infrastructure can impair users’ access to their applications. In addition, it speeds up access to credentials because there is no server latency issue.

Because Tivoli Access Manager for Enterprise Single Sign-On stores the local, secure credential file in a specific directory within the application data directory of the user profile. The file can be secured from other users by properly configuring Windows security on NTFS partitions. This also means that if Windows *roaming profiles* are enabled, users can log on to Windows from any computer within a domain and their credential file is available to them.

The Windows 2000 variable for the user profile directory is *%UserProfile%*, which defaults to *C:\Documents and Settings\%UserName%*. The file is stored as *%UserName%\AML.mdb* in *%UserProfile%\Application Data\IBM*, so the file for user *user1* might be as follows:

C:\Documents andSettings\user1\Application Data\IBM\user1\AML.mdb

15.1.5 Credential synchronization

While the agent stores user credentials and settings locally, it can synchronize the credentials and settings with remote file systems, directories, databases, and devices. Synchronization can be of the entire user database file (which contains all user credentials) or of individual records within the database. The

synchronization is triggered by a change to the Local Credential Store or settings. Synchronization can be extended to any storage mechanism through the *synchronization API*. Agent administration is fully supported through the *synchronization* component and allows the administrator to dynamically deliver updated settings and configuration data to the agent through the main storage mechanism. Synchronization can be of the entire credential file (Tivoli Access Manager for Enterprise Single Sign-On uses the newer of the local and remote files and overwrites the older ones through silent backup and restore functionality) or each individual credential record within the credential file (Tivoli Access Manager for Enterprise Single Sign-On uses the newer of the local and remote record for each record and overwrites the older ones).

Tivoli Access Manager for Enterprise Single Sign-On uses client-side intelligence in conjunction with a locally encrypted database of user credentials to respond to logon requests. Synchronizing user credentials to a directory service or network drive enables mobility, eases deployment, simplifies administration, and in certain settings (for example, public workstations) increases security.

Using LDAP, Tivoli Access Manager for Enterprise Single Sign-On supports multiple directory services including IBM Tivoli Directory Server, Sun ONE Directory Server, Novell NDS, and Microsoft Active Directory Server without modification. In addition, Tivoli Access Manager for Enterprise Single Sign-On supports a record level *file system synchronization*, *Windows roaming profiles*, file-level synchronization and backup, and provides a standard API for record-level synchronization of user credentials with any external application or device.

Figure 15-6 on page 462 shows an architectural overview.

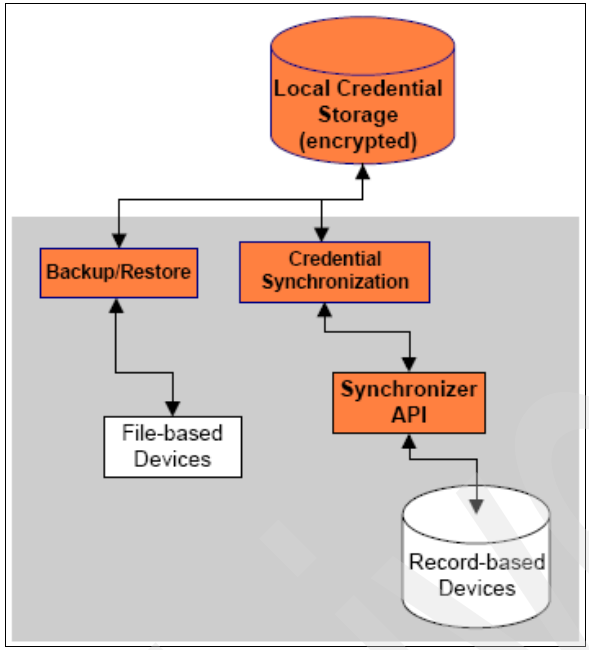


Figure 15-6 File credential synchronization

Tivoli Access Manager for Enterprise Single Sign-On can synchronize each user's credentials on a record-by-record basis with a remote store (for example, LDAP directory, Active Directory, NDS, a database, a file system, smartcard, and so on). Also, it can synchronize each user's credentials on a record-by-record basis (the record being their application credentials). Devices that store these records must have an extension to the synchronizer API.

As shown in Figure 15-7, the synchronizer API is a set of plug-in interfaces that the Tivoli Access Manager for Enterprise Single Sign-On *synchronization manager* uses to read and write data from and to the data source.

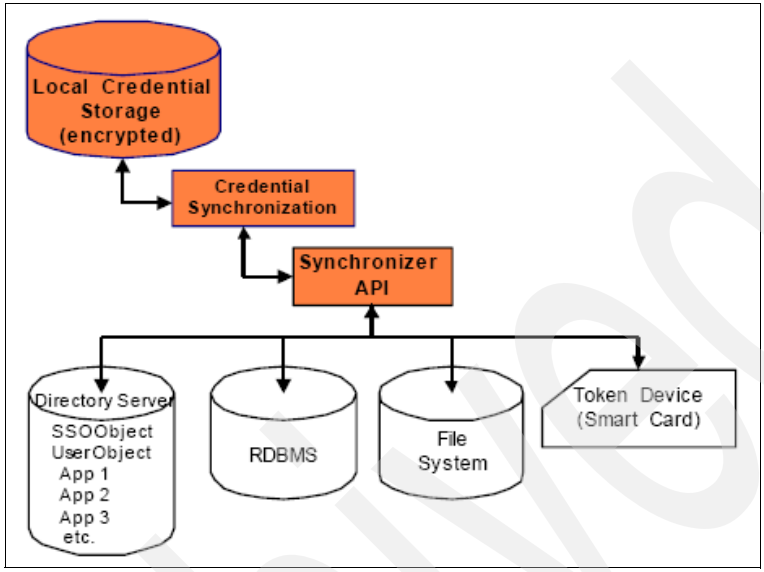


Figure 15-7 Record credential synchronization

The synchronization is based on record date and time and other authentication information. Put simply, Tivoli Access Manager for Enterprise Single Sign-On overwrites the older of the two local and remote records, replaces those files with the newer ones, and then uses the local credentials. The administrator can configure this process to occur as often as every credential change.

Note: This record-level synchronization enables users to utilize Tivoli Access Manager for Enterprise Single Sign-On from multiple computers in parallel.

If the remote store is a directory, file system, or database, users can access credentials from anywhere they have access to the store. If the remote store is a token, users can take the remote store with them.

Each storage device has its own requirements. The most common storage is in a directory server, such as LDAP, Active Directory, or Novell NDS as well as a file system on a server.

During synchronization, Tivoli Access Manager for Enterprise Single Sign-On takes credentials from the local credential file and credentials from remote storage, and merges them by date and time. If a set of credentials exists in one

place but not the other, it copies those credentials to the location where they are missing (either the AML file or remote storage). When a user deletes a set of credentials, Tivoli Access Manager for Enterprise Single Sign-On places that credential set's unique identifier (UID) in the UID list. Stored remotely, the UID list contains all deleted credential sets, each one containing a unique identifier.

Upon startup, Tivoli Access Manager for Enterprise Single Sign-On downloads the latest copies of the first-time use settings, application configurations, and administrative override settings, overwriting older versions. See *Overriding Settings: Registry Values* in the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console Help file for more information about these files including a complete list of their variables and values.[2]

Tivoli Access Manager for Enterprise Single Sign-On includes the following synchronizer extensions:

- LDAP-compliant directory servers
- IBM Tivoli Directory Server
- Oracle Directory Server
- Novell eDirectory
- Sun Java System Directory Server 5.1
- Critical Path Directory Server
- OpenLDAP Directory server
- Microsoft Active Directory Server
- SQL-compliant relational database systems
- Microsoft SQL Server
- IBM DB2
- Oracle 9i and 10g

Note: Tivoli Access Manager for Enterprise Single Sign-On also includes a synchronizer extension supporting a file system, such as on a remote network drive share.

You determine which synchronization modules to install on each computer, which modules to enable for each user, and how to configure each extension. By default the synchronizer module is installed without any additional synchronization extensions.

15.1.6 Event logging

Tivoli Access Manager for Enterprise Single Sign-On can report events locally and remotely. It can log all events, including credential use, credential changes,

[2] This help file is installed automatically with the Tivoli Access Manager for Enterprise Single Sign-On Administrative Console.

global credential events, Tivoli Access Manager for Enterprise Single Sign-On events, and Tivoli Access Manager for Enterprise Single Sign-On feature use. The solution can log the fields that administrators specify. As depicted in Figure 15-8, Tivoli Access Manager for Enterprise Single Sign-On can log events locally or to any external destination through the Event Logging API.

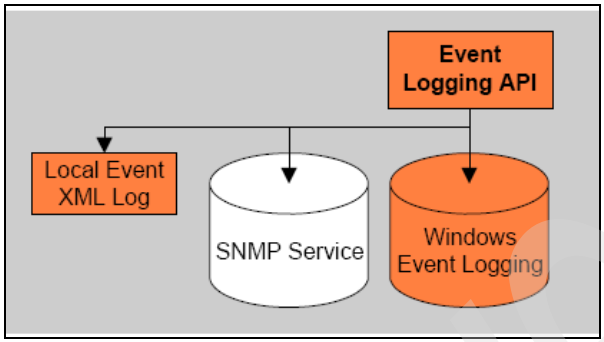


Figure 15-8 Event logging process

Specifically, Tivoli Access Manager for Enterprise Single Sign-On can log the following:

- Credential use events: log ins, manual password changes, automatic password changes.
- Credential changes: add credentials, delete credentials, change credentials, copy credentials.
- Global credential events: backup, restore, synchronize.
- Tivoli Access Manager for Enterprise Single Sign-On events: startup, shutdown.
- Tivoli Access Manager for Enterprise Single Sign-On feature use: Logon Manager, Settings, Help, About, and so on.
- Administrator-specified fields: Domain, Windows user name, system user name, Application name, Application user name, Application third field, Date, Time, and so on.
- Events can be logged to any desired destination: Local XML storage, SNMP service, Windows event log, or directory server.

Note: Default: No Event Logging modules are installed, and no logging occurs.

15.1.7 Additional components

Tivoli Access Manager for Enterprise Single Sign-On also includes the following miscellaneous modules:

- Screen saver

  Tivoli Access Manager for Enterprise Single Sign-On supplies a secure screen saver that works with the Windows 95, 98, and ME operating systems (systems that lack secure screen savers of their own). This screen saver requires a password and does not allow the user to bypass the password request.

- Backup/restore

  For users who do not perform any credential synchronization, the backup/restore component enables archiving and restoration of user credentials.

- Citrix/Terminal Services tools

  For environments that require usage of Tivoli Access Manager for Enterprise Single Sign-On within a Citrix or Windows Terminal Services environment, additional components are supplied to allow Tivoli Access Manager for Enterprise Single Sign-On to interact appropriately within each session.

- Installer Package

  Tivoli Access Manager for Enterprise Single Sign-On ships within a Windows Installer package that supports the flexibility of that technology for easier deployment and customization.

SSO File Sync Service

For servers sharing file systems for synchronization, the SSO File Sync Service ensures that proper rights are set on all user object trees.

15.1.8 Desktop Password Reset Adapter

The Tivoli Access Manager for Enterprise Single Sign-On Desktop Password Reset Adapter enables users to reset their Windows password from locked workstations and helps to eliminate costs that are associated with help-desk calls related to Windows password resets (Figure 15-9).

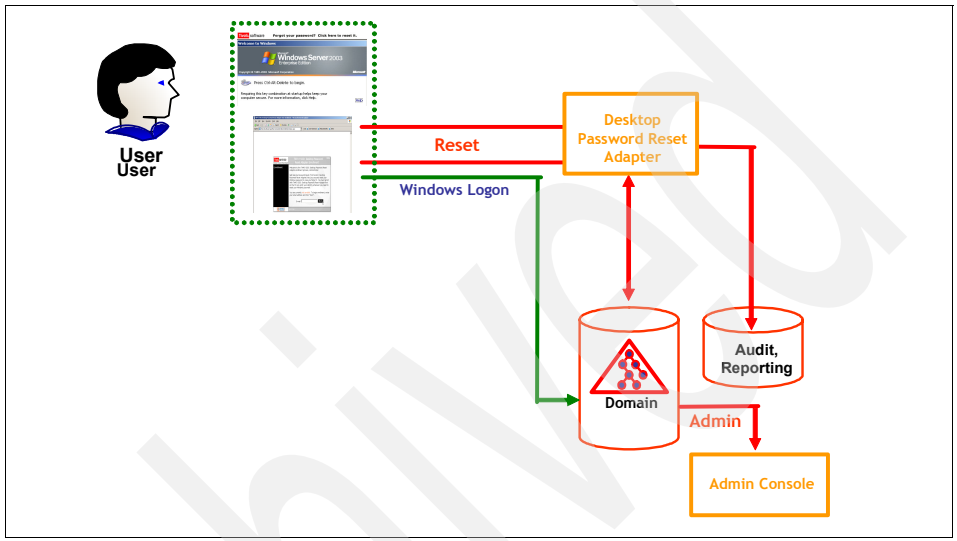


Figure 15-9 Desktop Password Reset Adapter Architecture

Desktop Password Reset Adapter should be integrated where the Windows Password is needed most and often forgotten—at the Windows Logon—to increase the likelihood of its access and use (Figure 15-10).

Figure 15-10 Windows Logon

The password reset process is Web browser based. It provides access on kiosks or from other machines when needed and can be integrated with other Web self-service mechanisms such as Tivoli Access Manager for e-business.

Note: The Desktop Password Reset Adapter resets the Windows Domain password only. It does *not* require access to a separate logged on computer.

The Tivoli Access Manager for Enterprise Single Sign-On Desktop Password Reset Adapter (DPRA) enables users to reset their primary authentication (Windows) password from a locked workstation based on a challenge-response process. All questions are customizable and configurable. When Tivoli Access Manager for Enterprise Single Sign-On DPRA is installed, users enroll by answering a series of confidential questions. When users forget their Windows password, Tivoli Access Manager for Enterprise Single Sign-On DPRA prompts the user to answer the questions again. An identity validation process compares the answers with the originals and factors in accounting for human errors in typing and memory recall (confidence-based authentication). If the user answers a sufficient number of questions successfully, the DPRA enables them to reset their Windows password automatically, and no call to the help desk is necessary.

Tivoli Access Manager for Enterprise Single Sign-On DPRA can call either the Tivoli Identity Manager password reset mechanism or the stand-alone IIS-based DPRA server. If Tivoli Identity Manager is chosen, then when Tivoli Identity Manager creates a new password the DPRA resets the user's password on the domain and closes the browser window, returning the user to the Windows sign-on window where they can sign in.

15.1.9 Authentication Adapter

The Authentication Adapter allows strong authentication using tokens, smart cards, proximity cards, and biometrics, as well as flexible authentication options such as stepping up from passwords to stronger authentication mechanisms for accessing select, critical resources. Figure 15-11 depicts the architecture for the Authentication Adapter.

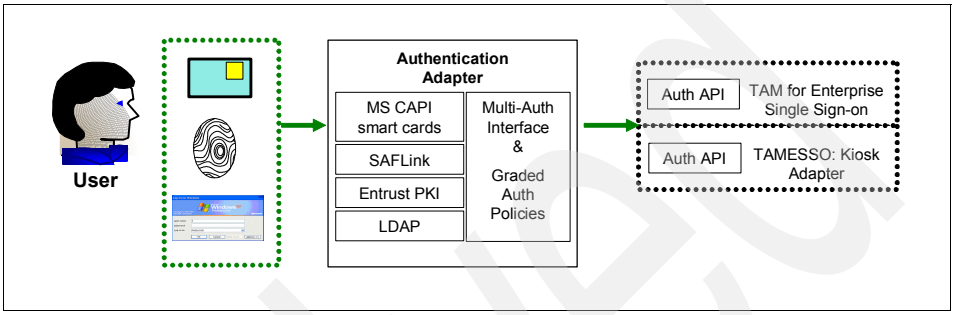


Figure 15-11 Authentication Adapter architecture

The Authentication Adapter enables organizations to bridge strong authentication to all of their applications seamlessly. Users can employ different authenticators at different times, and application access can be controlled based upon the authenticator used. The Authentication Adapter adds the following three capabilities:

1. Strong authentication support from a variety of strong authenticators, including smart cards and biometrics devices, for all authentication events like initial authentication, re-authentication, and forced authentication.
2. Multiple authenticator support allows multiple logon methods to be used to authenticate a user and provides an authenticator that is capable of supporting *graded authentication* as well as alternative logon methods. This allows users the ability to mix and match multiple logon methods on-the-fly.
3. Administrators can define grades or levels to authentication methods and to applications. This provides the ability to control what functions of the Authentication Adapter users can execute based upon the type of authenticator presented.

Note: Authentication Adapter files and components are installed directly into the Tivoli Access Manager for Enterprise Single Sign-On directory. A separate Authentication Adapter directory does not exist. Because the Authentication Adapter is an add-on module to Tivoli Access Manager for Enterprise Single Sign-On, Authentication Adapter Help is part of the Tivoli Access Manager for Enterprise Single Sign-On Help Documentation.

Multiple authenticator support

Multiple authentication supports the use of multiple logon methods to authenticate a user. This feature provides an authenticator that is capable of supporting graded authentication as well as alternative authentication methods.

Tivoli Access Manager for Enterprise Single Sign-On Authentication Adapter's multiple authenticator provides the following capabilities:

- Accepts authentication using different authenticators. It also supports *graded authentication*.
- Allows multiple authenticators to be used interchangeably during a user session, for example, between the initial logon and the logout.
- Allows multiple authenticators to be used interchangeably between sessions.
- Provides administrators the ability to do the following:
  - Allow or disallow the use of multiple authenticators.
  - Specify which authenticator is the default primary authenticator.
  - Specify which authenticators are required for enrollment.
  - Restrict access to applications based upon the strength of the authenticator used.
  - Allow or disallow the use of multiple authenticators interchangeably during a single session.
  - Allow or disallow the use of multiple authenticators interchangeably between sessions.

Graded authentication

Graded authentication lets you define *grades* or *levels* to authenticate in the Authentication Adapter. Graded authentication controls what functions of the Authentication Adapter users can execute based upon the type of authenticator presented. Levels, or grades, can be applied and used to ensure the correct level of authentication is performed for specific events or activities.

Tivoli Access Manager for Enterprise Single Sign-On Authentication Adapter's graded authentication supports the following capabilities:

- An unbounded number of authentication grades or levels.
- Setting required authentication grades on a per application basis.
- Setting required authentication grades on Tivoli Access Manager for Enterprise Single Sign-On processes that require re-authentication.
- Administration set up for the authentication level for every application.
- Administration set up for the authenticator grade.
- Logging of graded authentication events.
- Administration of the following:
  - Graded authentication support to be turned on or off.
  - Configuration of graded authentication on a per-application basis.

The Authentication Adapter controls application logins that can be initiated by the user, based upon the authenticator used by the user on the most recent authentication request. The most recent authentication request might be the initial logon, the last re-authentication, or the forced authentication requested by the Authentication Adapter.

The Authentication Adapter has an authentication grading scheme to which different authenticators are mapped and, separately, to which application logins are mapped. The Authentication Adapter only allows users to logon to an application when the grade of the authenticator used equals or exceeds that of the application logon.

When a user does not respond to an authentication request with an authenticator of sufficiently high grade, the Authentication Adapter prompts the user to either re-authenticate with an authenticator of sufficiently high grade or cancel the requested logon. If a user repeatedly attempts to initiate a logon or function with an authenticator of insufficient grade, the Authentication Adapter locks out the user, logs an event in the Event Manager, and notifies the user and administrator. If a user does not have the Authentication Adapter installed but their application logins were configured to require strong authentication, the user does not have access to those applications (for example, strong authentication is deployed in the enterprise, but not to that user).

The Logon Manager only displays the application logins that are currently available, based upon the authenticator used in the most recent authentication request.

You can configure the following Authentication Adapter functions to be accessible or inaccessible based upon the grade of authenticator that is used in the most recent authentication request:

- System Tray: Logon Manager
- Logon Manager: Delete, Properties, and Reveal All functions
- Logon Manager → Properties Page: Reveal Password function

If the Reveal All function is accessible based upon a grade of authentication used, it only reveals passwords for those applications whose grade is equal to or lower than the grade used to authenticate for that function.

15.1.10 Provisioning Adapter

The Provisioning Adapter automates the user credential distribution process so that identity management solutions such as Tivoli Identity Manager can provision and remove user involvement in the credential provisioning and management process. It enables an administrator to automatically provision Tivoli Access Manager for Enterprise Single Sign-On with a user's ID and password by using a provisioning system. An administrator is able to *add*, *modify*, and *delete* IDs and passwords for particular applications within the provisioning system and have the changes reflected in Tivoli Access Manager for Enterprise Single Sign-On. From the provisioning system, all user names and passwords inside of Tivoli Access Manager for Enterprise Single Sign-On can also be deleted so that a user's access to all protected applications is eliminated.

Figure 15-12 on page 473 illustrates the Provisioning Adapter architecture.

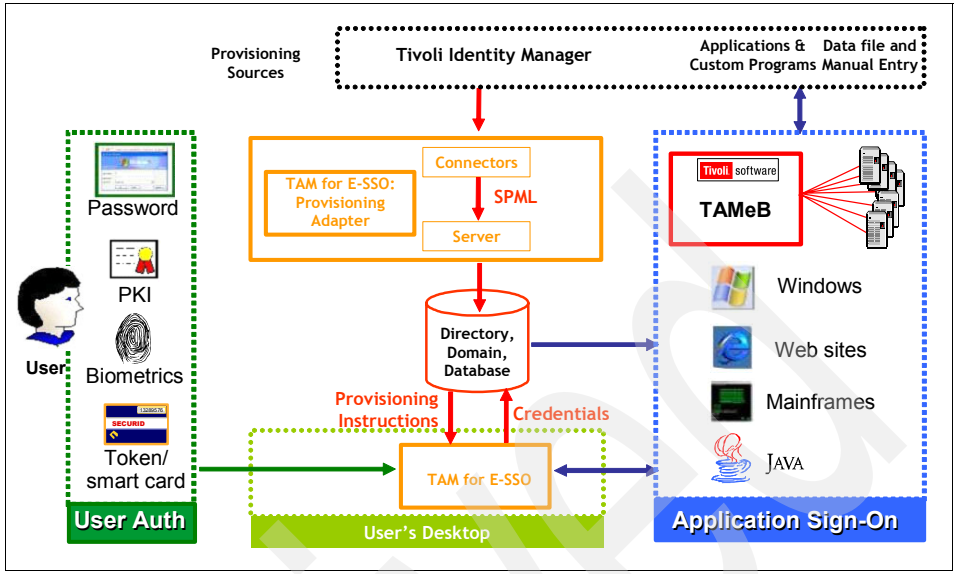


Figure 15-12 Provisioning Adapter architecture

The Provisioning Adapter can be implemented as a stand-alone management tool for credentials that are stored in a Tivoli Access Manager for Enterprise Single Sign-On implementation. It provides a Web-based interface that shows you the applications that are part of the Tivoli Access Manager for Enterprise Single Sign-On environment, and it allows the administrator to manually add, modify, or remove credentials. In addition to an administrative Web-based interface, it also provides a command line interface, Java interface, and includes logging and reporting tools.

In most organizations, users have to know, remember, and enter their application credentials. This is a particular hassle on the first day a user begins work or takes on a new set of responsibilities and permissions. But when an organization uses the Tivoli Access Manager for Enterprise Single Sign-On Provisioning Adapter, application credential provisioning and deprovisioning between Tivoli Identity Manager and Tivoli Access Manager for Enterprise Single Sign-On are automated. Consequently, organizations no longer need to physically distribute credentials to users who must enter them manually into Tivoli Access Manager for Enterprise Single Sign-On.

Instead, administrators directly create, edit, and delete user credentials through Tivoli Identity Manager. Users can enjoy single sign-on from day one and are no longer responsible for keeping track of their own application credentials. All while helping maximize security. When users no longer need access to systems, the integration between the Tivoli applications enables Tivoli Identity Manager to

remove the users' system and application access and also delete their credentials automatically from the Tivoli Access Manager for Enterprise Single Sign-On data store. Controlling the appropriate level of access helps maximize security and assists with compliance initiatives by demonstrating enforcement of internal controls to auditors.

Furthermore, Tivoli Access Manager for Enterprise Single Sign-On Provisioning Adapter provides a high level of administrative control. For example, when application passwords are reset in Tivoli Identity Manager, Tivoli Access Manager for Enterprise Single Sign-On is simultaneously updated so that it always has the correct password. Additionally, it extends audit and reporting capabilities to include information about applications and use of applications that are configured in Tivoli Access Manager for Enterprise Single Sign-On but that fall outside the Tivoli Identity Manager umbrella. Administrators can use the adapter to view a list of all users who are allowed to use a particular application. Or, conversely, they could see all the applications that a particular user can access.

The Provisioning Adapter receives instructions from Tivoli Identity Manager that contain credential data. It informs individual Tivoli Access Manager for Enterprise Single Sign-On agents about application configurations that were added, deleted, or changed by the following:

- Normalizing these instructions into a format that Tivoli Access Manager for Enterprise Single Sign-On can understand.
- Placing them into the directory object for the appropriate user.

When the Tivoli Access Manager for Enterprise Single Sign-On agent synchronizes with the database or directory, it reads and processes the instructions and then updates the entries as needed in its local credential cache. Depending on the instructions that it receives, the Tivoli Access Manager for Enterprise Single Sign-On agent might add, modify, or delete credentials in the appropriate user's local credential cache. Finally, the Tivoli Access Manager for Enterprise Single Sign-On agent synchronizes the credentials back to the database directory object for that user.

The Provisioning Adapter includes the following logical components:

- Server

  Accepts account credential provisioning information through a Web services interface. It also communicates that information to Tivoli Access Manager for Enterprise Single Sign-On agents by placing provisioning instructions into the directory or data store.

- Console

  Provides a Web-based administration GUI for communicating with the server.

- Command line interface

  Enables applications and administrators to communicate with the server.

- Connector

  Integrates the server and Tivoli Identity Manager through the CLI. The connector is a Java-based class library that is implemented as a workflow extension and can be incorporated into any Tivoli Identity Manager provisioning operation. Consequently, administrators can add, edit and delete application credentials for users through the Tivoli Identity Manager interface. The connector works on any platform where Tivoli Identity Manager runs.

Event logging and reporting

The Tivoli Access Manager for Enterprise Single Sign-On Provisioning Adapter contains an administrator-controlled event logging capability that enables organizations to monitor and record events.

The adapter can run a number of audit reports:

- All users that have a particular application configured in Tivoli Access Manager for Enterprise Single Sign-On
- All applications configured in Tivoli Access Manager for Enterprise Single Sign-On for particular user
- All provisioning requests
- Usage, based on user object detail

Note: To analyze the event log, simply export it as a comma-separated values file. Then import the file into virtually any tool that is used for analysis.

15.1.11 Kiosk Adapter

The Kiosk Adapter delivers a secure and user friendly solution that addresses the needs of traditional single logout in a kiosk environment and that is easy to maintain.

This solution provides user identification to the kiosk by prompting users to login with a Windows password or any supported primary authenticator. The Kiosk Adapter has a client-side agent that suspends or closes inactive sessions and seamlessly shuts down all applications.

Figure 15-13 on page 476 illustrates an architecture overview of the Kiosk Adapter.

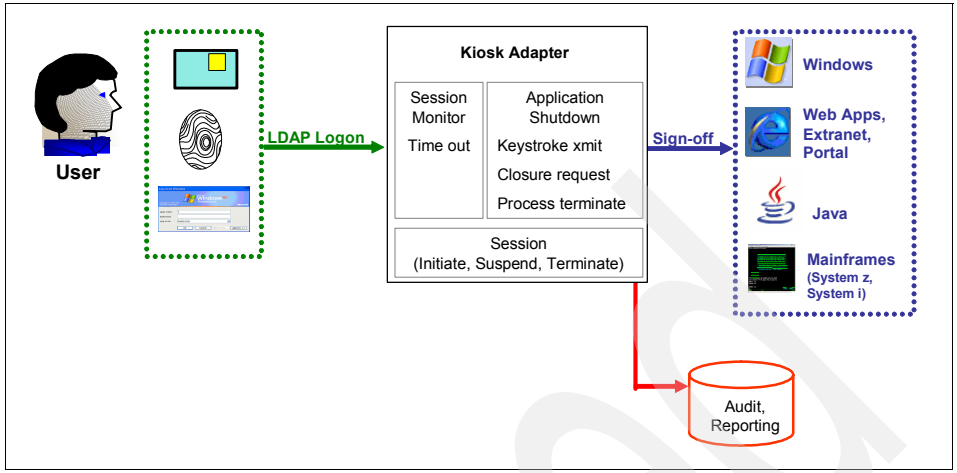


Figure 15-13 Kiosk Adapter architecture

The Kiosk Adapter adds the following capabilities:

- System logon

  Two modes of system logon are supported:

  - Automatic

    When the kiosk boots up, it automatically logs on to a generic user account, and all subsequent logins/logouts into Windows are disabled.

  - Manual

    When the kiosk boots, it prompts the user to log in.

- Session Suspend and Un-suspend

  A session is suspended upon either of the two following events:

  - Current session is inactive for a predefined period of time.
  - User logs out of current session.

  Note: A session is resumed when the user re-authenticates to the suspended session.

- Session logoff

  A suspended session is automatically logged off upon either of the two following events:

  - The session was suspended for a predefined period of time
  - A new user initiates a new session at the kiosk

Applications can be closed using multiple methods, including the following:

– Transmission of key stroke sequences to the application
– Window messages (application closure requests)
– Process termination

15.2 Physical component architecture

In this section, we describe the physical components that are assembled for Tivoli Access Manager for Enterprise Single Sign-On as well as some step-by-step walkthroughs. Figure 15-14 shows a simple, base deployment architecture.

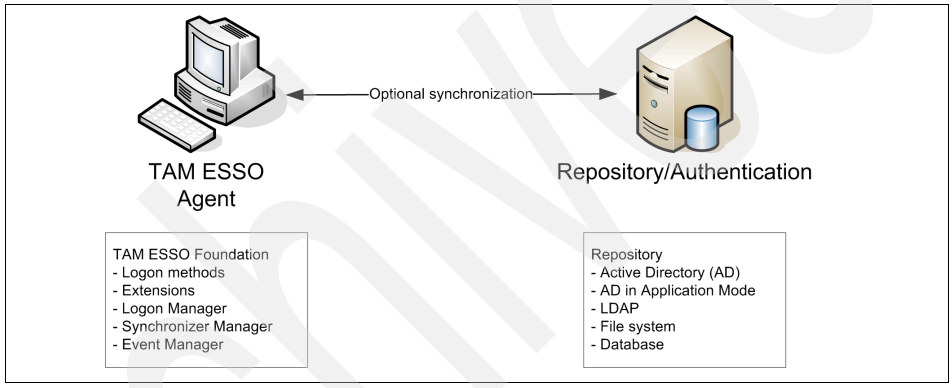


Figure 15-14 Physical base deployment architecture

15.2.1 Agent

The Tivoli Access Manager for Enterprise Single Sign-On agent foundation gets deployed on user workstations either manually or using software distribution mechanisms. With this agent foundation, several configuration options can be deployed:

- Logon methods

  The logon methods are plug-ins that provide different methods for logging onto Tivoli Access Manager for Enterprise Single Sign-On. By default, Windows Logon is installed.

Note: If you are using a server-based instead of the local Windows login mechanism to authenticate to Tivoli Access Manager for Enterprise Single Sign-On, you must ensure that the adequate server (Active Directory, LDAP, and so on) is available.

The plug-ins available are as follows:

- Windows Logon

  Plug-in that enables logging onto TAM E-SSO by logon to Windows.

- Windows Logon V2

  Plug-in that enables logging onto TAM E-SSO by logon to Windows. This plug-in also includes secure pass phrase and Graphical Identification and Authentication (GINA) DLL support.

- GINA

  GINA module that works with the Windows Domain logon method.

- LDAP

  Plug-in that enables logging onto TAM E-SSO by logon to an LDAP directory.

- LDAP V2

  Plug-in that enables logging onto TAM E-SSO by logon to an LDAP directory. This plug-in also includes secure pass phrase support.

- Authentication Manager

  This feature adds the capability to allow multiple logon methods to authenticate the user.

- Extensions

  The extensions are plug-ins that enhance and extend functionality of Tivoli Access Manager for Enterprise Single Sign-On. By default, the Backup/Restore Manager, Logon Manager, and Setup Manager are installed. The plug-ins available are as follows:

  - Backup/Restore Manager

    This plug-in provides a simple file-based backup and restore mechanism through a wizard interface.

  - Logon Manager

    This plug-in provides the main credential management, request, and delivery interfaces.

- Setup Manager

  This plug-in provides the initial first time user experience when setting up the application.

- Synchronization Manager

  This plug-in provides for the management of synchronization extensions to the application.

- Event Manager

  This plug-in provides for the management of event logging extensions to the application.

- Logon Manager

  There are several helper plug-ins available that assist with SSO.

  - Internet Explorer Helper

    Extension helpers that add SSO support for Internet Explorer.

  - Mozilla Browser Helper

    Extension helpers that add SSO support for Mozilla-based browser.

  - Mainframe Emulator Helper

    Extension helpers that add SSO support for HLLAPI-based emulators. The Mainframe helper extensions are as follows:

    - Console Window Support

      Support for Console windows (command prompt) within Tivoli Access Manager for Enterprise Single Sign-On's mainframe plug-in.

    - Legacy Emulator Support

      Support for 16-bit existing HLLAPI-based emulators.

  - Java Helper

    Extension helpers that add SSO support for Java applications natively.

  - SAP Helper

    Extension helpers that add SSO support for SAP applications.

    In order for this to work SAP must be configured to work with Tivoli Access Manager for Enterprise Single Sign-On. See the Technical Notes in *IBM Tivoli Access Manager for Enterprise Single Sign-On Release Notes Version 6.0*, SC32-1990.

- Synchronizer Manager

  The synchronizer plug-ins available are as follows:

  - Active Directory Synchronizer

    Synchronization plug-in that supports storage and retrieval of credentials and settings from an Active Directory server.

  - LDAP Synchronizer

    Plug-in that supports storage and retrieval of credentials and settings from an LDAP-compliant directory, such as Tivoli Directory Server or Sun Java System Directory Server.

  - ADAM Synchronizer

    Synchronization plug-in that supports storage and retrieval of credentials and settings from a Microsoft Active Directory or Active Directory Application Mode (ADAM) server.

  - File System Synchronizer

    Synchronization plug-in that supports storage and retrieval of credentials and settings from a file share.

  - Database Synchronizer

    Synchronization plug-in that supports storage and retrieval of credentials and settings from a database.

  - Roaming Profile Synchronizer

    Synchronization plug-in that supports roaming profiles.

- Event Manager

  The plug-ins available are as follows:

  - XML File

    Event Management plug-in that supports logging of events to a local XML file.

  - Windows Event Extension

    Event Management plug-in that supports logging of events to the Windows Event Manager.

15.2.2 Repository and authentication

By default, the agent uses a locally encrypted credential store for mobile or offline use. The other component in the base architecture is the Tivoli Access Manager for Enterprise Single Sign-On repository. The repository can be used as a centralized encrypted storage for user credentials and agent configuration.

The client can back up and restore credentials to and from the central repository, automatically using synchronizer plug-ins as described previously. Several different forms of repository are supported, including the following:

- Sun ONE Directory Server
- Microsoft Active Directory
- Microsoft ADAM (Active Directory in Application Mode)
- Novell eDirectory
- Microsoft SQL
- Oracle
- IBM DB2
- Network Drive Shares
- Any v2/v3 LDAP directory

In order to use any of these repositories individual configurations have to be performed.

Tivoli Access Manager for Enterprise Single Sign-On can also be configured to have the users log on to a central authentication server like Active Directory or an LDAP server.

15.2.3 Administrative Console

In order to perform centralized administration for Tivoli Access Manager for Enterprise Single Sign-On you deploy the Administrative Console, as shown in Figure 15-15.

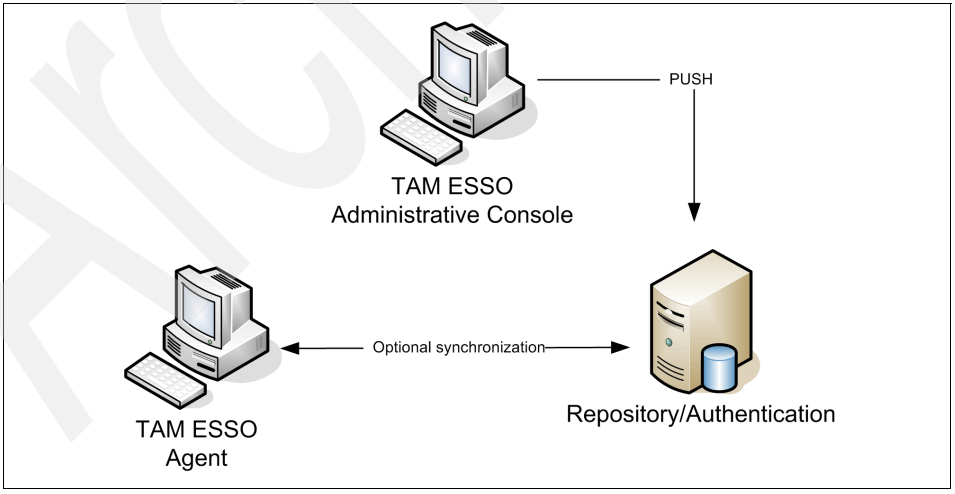


Figure 15-15 Administrative Console deployment

The Administrative Console enables both agent and server configuration of most agent options. All changes are being pushed to the central repository and are then synchronized back to the agents. The Administrative Console enables the following configuration options:

- Easy creation, management, and deployment of the following:
  - Application configurations and application configuration lists
  - Credential-Sharing Groups
  - Password Policies
  - Bulk-add lists
  - Agent configuration settings (through registry settings)
- Easy set up and management of synchronizer extensions:
  - LDAP Directory Servers, including Tivoli Directory Server, Novell eDirectory, Oracle Directory Server, Sun Java System Directory Server 5.1, Critical Path Directory Server, And OpenLDAP Directory Server.
  - Microsoft Active Directory Server systems (including Application Mode)
  - Relational database systems, including Microsoft SQL Server, IBM DB2, and Oracle 9i/10g
  - File systems

The Administrative Console obsoletes the need for editing configuration files or the registry by hand with the associated risks of errors or providing invalid parameters.

Initial deployment scenario

Now it is time to take a closer look at a possible initial deployment scenario for Tivoli Access Manager for Enterprise Single Sign-On. Figure 15-16 on page 483 shows a numbered layout picture.

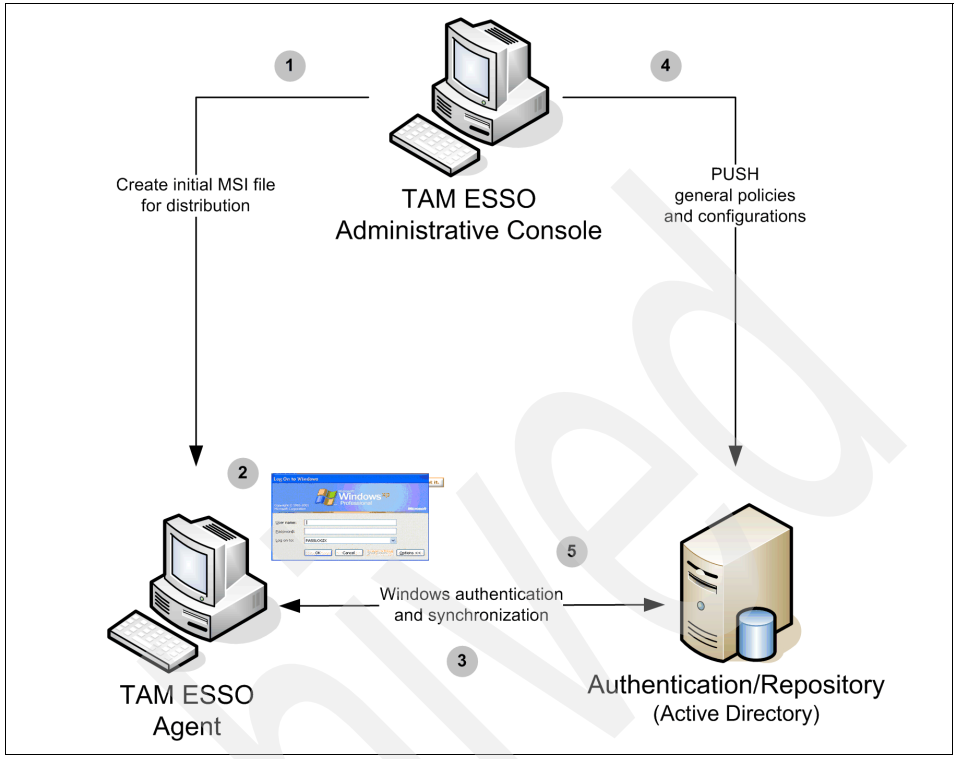


Figure 15-16 Initial deployment overview

Following is a walk through of the steps illustrated in Figure 15-16:

1. The administrator installs the Tivoli Access Manager for Enterprise Single Sign-On agent and the Administrative Console and prepares a Microsoft installer (MSI) file that includes the initial configuration for each desktop agent.

   The administrator then uses the Administrative Console to configure applications and other options. In this case, the user logon module uses the standard Windows login mechanism, and the synchronizer manager is configured to replicate Tivoli Access Manager for Enterprise Single Sign-On data with the central repository based on Active Directory.

   When the configuration of the MSI file is completed, the administrator can deploy it to the enterprise workstations (for example, using an existing software provisioning mechanism).

2. The user authenticates to the central Active Directory using the standard Windows login mechanism.

   After the Tivoli Access Manager for Enterprise Single Sign-On agent is deployed and started for the first time, the user is prompted to provide some additional information for the preconfigured applications, for example, a password.

3. After providing this information, and every time the user changes a Tivoli Access Manager for Enterprise Single Sign-On setting on the local workstation, the agent encrypts the data and stores it locally as well as in the central repository on Active Directory.

   From this point forward, the user is operating as usual. However, the user is no longer required to provide authentication information other than the initial Windows user ID and password.

   In the event of a password update request from a particular application, the Tivoli Access Manager for Enterprise Single Sign-On agent can generate a new password according to the password policy. It then sends the new password to the application, stores it on the local workstation, and synchronizes it with the central repository. When settled in this password renewal mechanism, there is no further need for users to remember a password.

   Note: Automatically generated random passwords do not have to be remembered by the user. In case of curiosity or other reasons, the password can, however, be displayed in clear text on the user's workstation.

4. After the initial deployment, the administrator needs to add new Tivoli Access Manager for Enterprise Single Sign-On profile information for additional applications and other changes, for example, an updated password policy.

   The administrator then uses the Administrative Console to push the changes into the central repository.

5. After these changes are published in the repository, the Tivoli Access Manager for Enterprise Single Sign-On agents pick up the changes and synchronize with the local profiles that are stored on the individual workstations.

15.2.4 Authentication Adapter

The Authentication Adapter enables organizations to bridge strong authentication seamlessly to all of their applications, including smart cards, biometric devices, and Entrust authenticators. Users can employ different

authenticators at different times and application access can be controlled based upon the authenticator used.

The physical components needed to implement strong authentication, like smart card readers, biometrical scanners, and so on, must be deployed at the agent. In addition to the physical components you also have to deploy the Authentication Adapter add-on module for the Tivoli Access Manager for Enterprise Single Sign-On agent.

If you deploy an application that requires strong authentication, you have to make sure that workstations for the particular users are equipped with the proper authentication mechanism and hardware. Otherwise, the user cannot authenticate towards that particular application.

Note: Most strong authentication devices that you can deploy for your clients might have their own requirements in terms of additional components. Make sure you have satisfied all those requirements before deploying the solution.

15.2.5 Kiosk Adapter

The Kiosk Adapter delivers a secure and easy to use administer solution that addresses the needs of traditional single sign-off in a kiosk environment. This solution provides identification to the kiosk by prompting users to login with a Windows password or any supported primary authenticator. The Kiosk Adapter has a client-side agent that suspends or closes inactive sessions and seamlessly shuts down all applications.

There are no additional physical components required to deploy a Kiosk Adapter onto a Tivoli Access Manager for Enterprise Single Sign-On agent. For more information about setting up the Kiosk Adapter refer to *IBM Tivoli Access Manager for Enterprise Single Sign-On Kiosk Adapter Installation and Setup Guide Version 6.0*, SC32-1997.

15.2.6 Desktop Password Reset Adapter

The Desktop Password Reset Adapter (DPRA) lets you access your Windows user account when you lose or forget your password. There is no need to call your help desk or technical support and no waiting for an administrator to reset your password.

Instead, you have to do a quick *pop-quiz* that verifies that you are really *you*, and you can reset your password yourself. If you are the account owner, you should

always pass because you created the quiz answers when you completed the DPRA enrollment interview.

To deploy the DPRA in your environment, you have to provide two additional physical components, as shown in Figure 15-17.

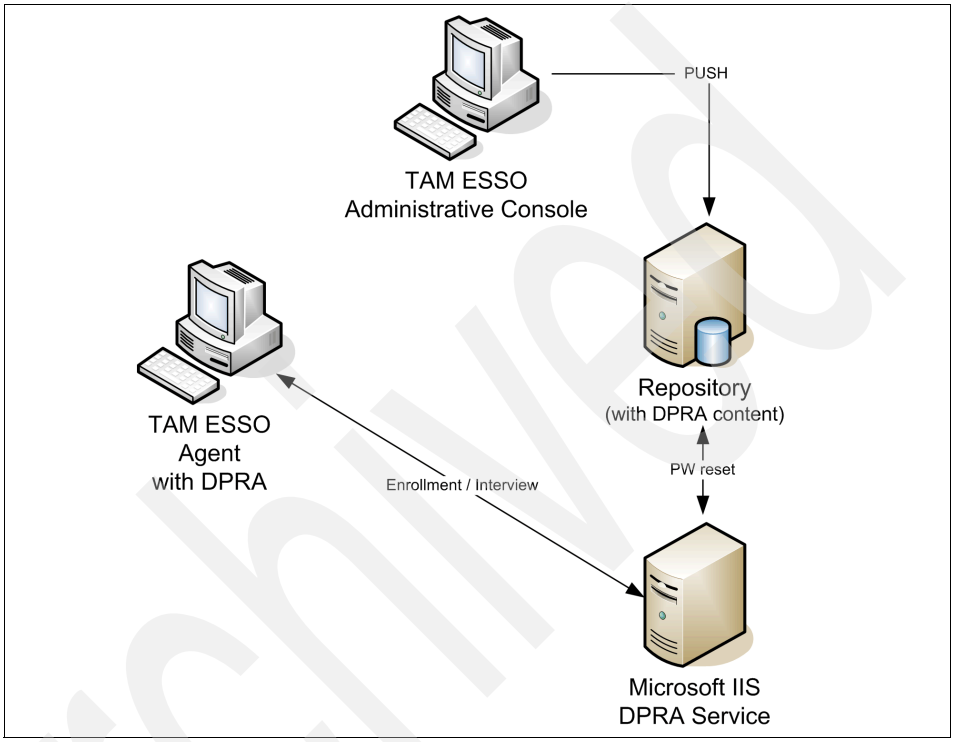


Figure 15-17 DPRA physical architecture

The first component necessary is the DPRA Service. This component provides the Web Service (Microsoft IIS/.NET based) interface for the agent to interact with when the user has to reset his password or when he initially enrolls for the password reset service. In order for this connection to be established you also have to install the DPRA client-side agent.

The second component provides the storage capability for the DPRA interview questions, the enrolled users, and their answers, which can be stored in either one of the following:

- Microsoft Active Directory or Active Directory Application Mode (ADAM)
- Microsoft SQL Server 2000
- Oracle Database

Because you have to use Active Directory or ADAM as your authentication repository with the DPRA service, you will probably also use that component to store your DPRA data. However, you can decide to store the data on any of the supported platforms. You can also have those services (SQL Server or Oracle) run on their own physical machine as shown in Figure 15-18.

Note: Be advised that you should not consider deploying both the IIS services and your primary Active Directory on the same physical machine in a production environment.

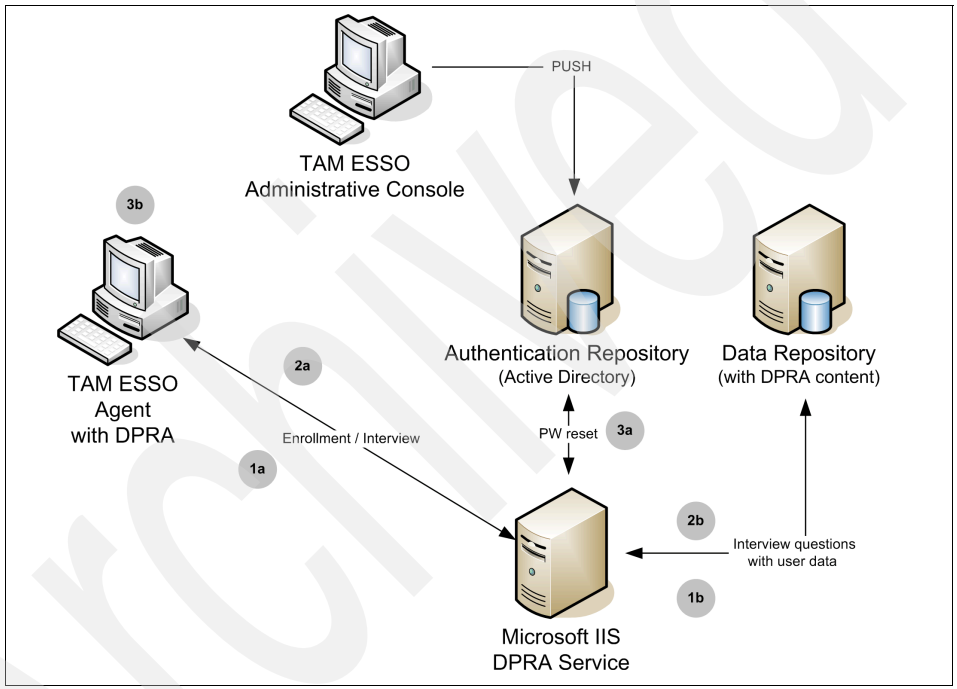


Figure 15-18 DPRA physical architecture - continued

Following is a walk through of the single steps involved in using the DPRA, as illustrated in Figure 15-18:

1. After the DPRA component is purchased and deployed on the workstations, the user has to perform an initial enrollment using a Web application provided by the DPRA Service (1a). A challenge/response set of questions are being filled in and stored in the DPRA data repository (1b).
2. After the Windows workstation logon password is reset, the user then uses the special link on top of the Windows logon dialog as shown in Figure 15-19 on page 488 (2a).

Figure 15-19 Initiating the Windows password reset

After walking through a Web-based challenge or response quiz (2b) successfully, the user provides a new password.

3. The DPRA Service finally updates the password on the Active Directory server (3a), and the user can log on to the workstation by providing the new password (3b).

15.2.7 Provisioning Adapter

The Provisioning Adapter Server can receive and process provisioning requests initiated by Tivoli Identity Manager. The integration between the Provisioning Adapter Server and Tivoli Identity Manager is accomplished by using a workflow extension that Tivoli Identity Manager uses to communicate with the Provisioning Adapter Server Web Service.

Figure 15-20 on page 489 illustrates the necessary physical components.

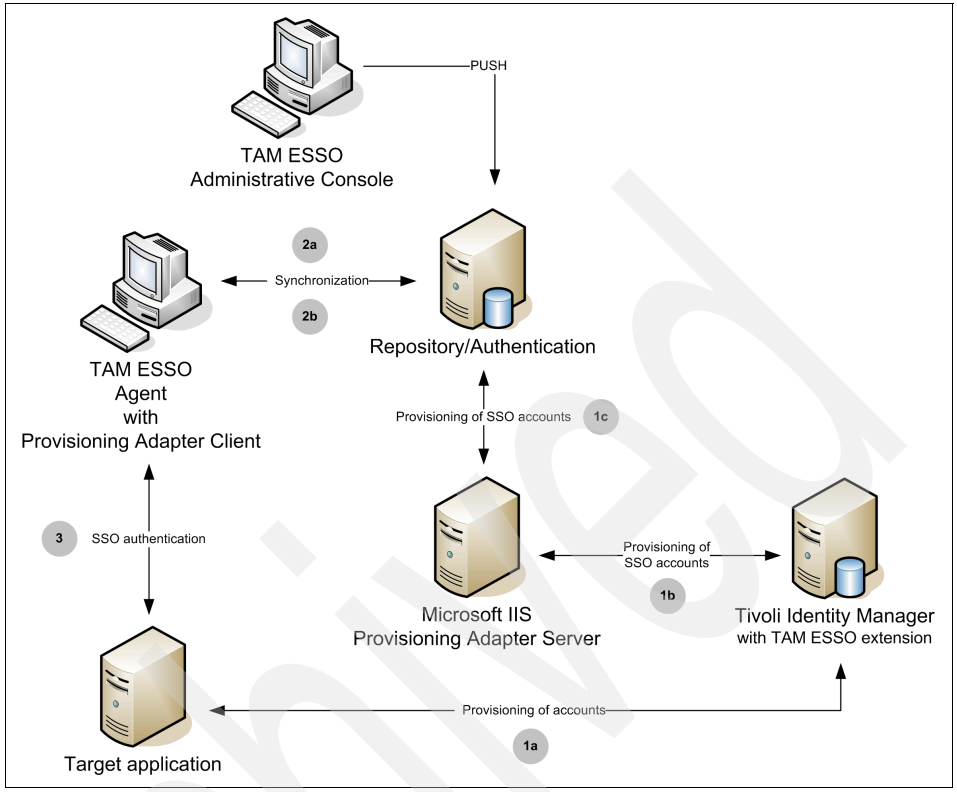


Figure 15-20 Provisioning Adapter physical architecture

The best way of describing the individual components is by taking the following step-by-step walkthrough:

1. Tivoli Identity Manager is responsible for centrally provisioning accounts and maintaining the overall user lifecycle management, including modifications and deletions. In our scenario, the *Tivoli Access Manager for Enterprise Single Sign-On workflow extensions* are installed and configured on the Tivoli Identity Manager system.

   Let us assume that our user obtained a new job role and needs access to a specific target application. Tivoli Identity Manager picks up the role change for the user automatically and provisions a new user account and password according to the centralized policies (1a).

   Due to the workflow extensions and the fact that the target application is configured to use them, Tivoli Identity Manager also initiates a provisioning call to the Provisioning Adapter Server (1b), which takes over the responsibility to create the SSO user account for that particular application in the Tivoli Access Manager for Enterprise Single Sign-On repository (1c).

When this provisioning step takes place, the new SSO user account information is placed in a special staging area for that individual user where it awaits further processing.

Note: At this point, the new application is fully available to the user without the user ever knowing the provisioned password. Although the following step is necessary for the provisioning process, the user can access the application immediately after logging into the Windows workstation.

As mentioned previously, out of curiosity the user could use the Tivoli Access Manager for Enterprise Single Sign-On Logon Manager on the workstation to reveal the new password, although the user will never need to do this.

2. When the user logs into the Windows workstation, the Tivoli Access Manager for Enterprise Single Sign-On agent picks up the information in the staging area (2a), encrypts the password with the local key, and stores the password again in the local store and on the Tivoli Access Manager for Enterprise Single Sign-On repository (2b). Upon completion the information in the staging area is removed.
3. After the user logs into the Windows workstation, the user can immediately invoke the new application without providing a password.

15.3 Conclusion

This concludes the architecture and component design for Tivoli Access Manager for Enterprise Single Sign-On. We took a close look at the internal data flow between the different logical components and the optionally available adapters. By using physical component diagrams for the different scenarios, we provided descriptions on how to deploy the different physical components as well as step-by-step workflows.

16

Tivoli Access Manager for Enterprise Single Sign-On scenario

In this chapter we describe the architecture components for deploying Tivoli Access Manager for Enterprise Single Sign-On within an example configuration.

Throughout this chapter, we provide a real world example of an enterprise, we define the business and functional requirements, and give a detailed account on the architecture for an Enterprise Single Sign-On Solution.

16.1 Company profile

In this chapter we introduce Areally Big Investment Corporation, an investment bank that is headquartered in a large metropolitan area. As with all financial institutions, it has a diverse set of business drivers and operating environments. Areally Big Investment Corp. offers broker services to their customers and handles transactions for the various stock markets around the world. With offices located around the globe, Areally Big Investment Corp. has 5,000 employees who have access to financial data systems, e-mail, and other intranet applications.

There are several smaller data centers and help desk facilities located around the world but the main data center and help desk facilities are located within a one-day drive of the company headquarters. Areally Big Investment Corp. currently has a robust security and network infrastructure in place. These systems are audited regularly, and risk assessments are scheduled quarterly.

Because there are numerous operating regulations that have been imposed by various government authorities, Areally Big Investment Corp. is implementing a worldwide security policy that supports these regulations.

However, Areally Big Investment Corp. has been aware for some time that there are some problems that need to be addressed in the password management area.

16.2 Current IT Architecture

Areally Big Investment Corp. currently has a distributed architecture based on different banking applications, as well as other related technologies. Figure 16-1 on page 493 is a diagram of the logical architecture. The diagram is not following the usual networked layout standard because it is meant to only show the logical components. The physical topology is not of any consequence to this scenario.

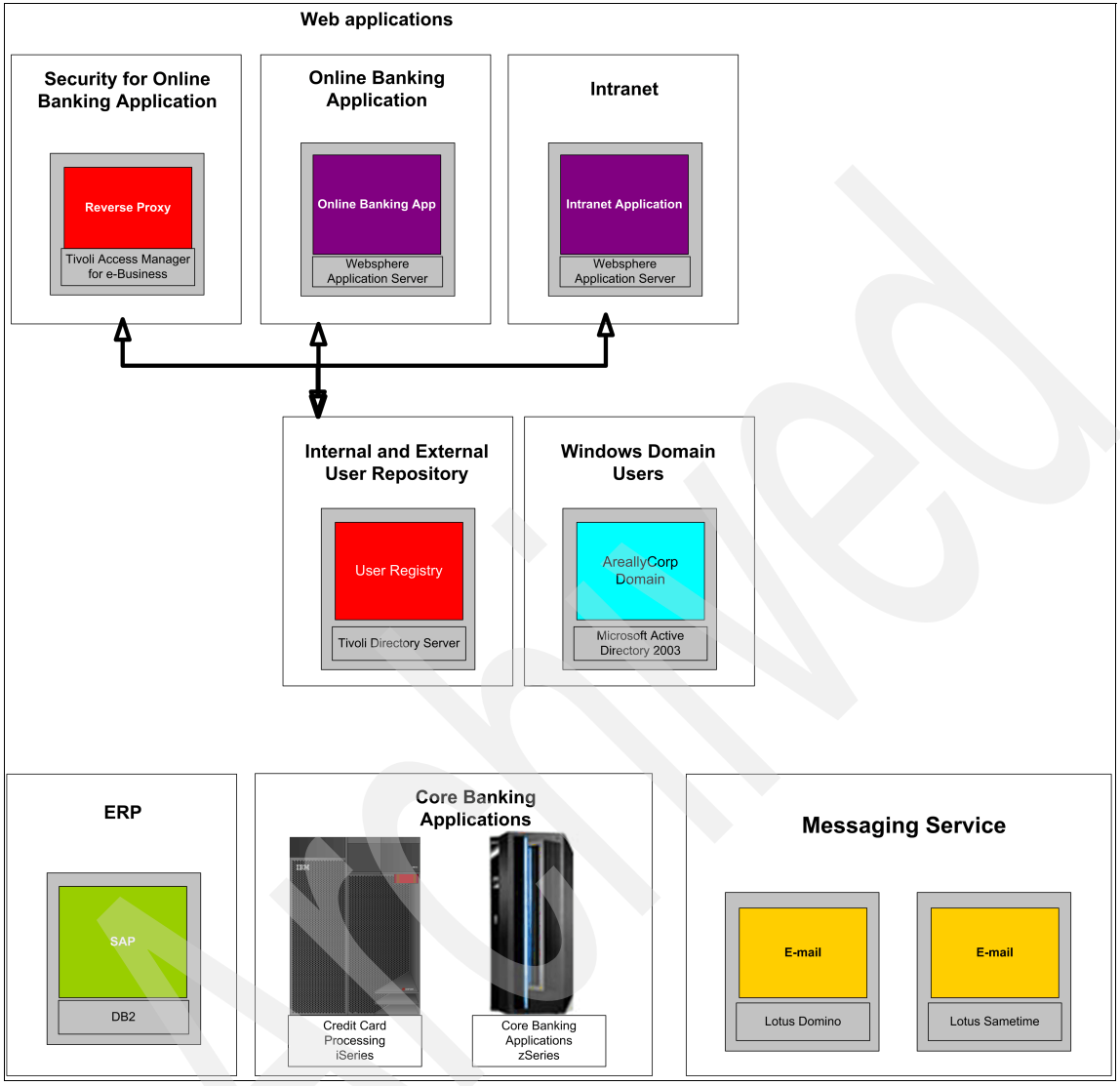


Figure 16-1 Logical architecture of Areally Big Investment Corporation

The corporation's main banking applications run on an IBM System z mainframe. Most of the employees use this application for their work. The credit card processing software runs on an IBM System i, and it is used by the credit card teams. In the corporation's intranet a J2EE application is deployed on WebSphere Application Server. The online banking application also runs on another instance of a WebSphere Application Server, and it is protected using Tivoli Access Manager for e-business.

The corporation's ERP is based on SAP with a DB2 back-end. For corporate messaging the standard is Lotus Domino, and for corporate instant messaging the standard is Lotus Sametime®.

As for directory technologies, the main corporate directory is based on Tivoli Directory Server, which hosts the internal and external user directories. There is also a Microsoft Windows Active Directory for domain authentication in the Windows environment.

Important: The purpose of this chapter is to demonstrate a scenario for Tivoli Access Manager for Enterprise Single Sign-On. The Tivoli Access Manager for e-business and Tivoli Directory Server components presented here are there for architectural reasons. For information about a scenario focused on Tivoli Access Manager for e-business see Chapter 7, "A basic WebSEAL scenario" on page 245.

For the purpose of this scenario, we are interested in the authentication to the different environments to be able to automate this process with the implementation of an enterprise single sign-on solution. Table 16-1 shows the different client applications for the previously explained architecture:

Table 16-1 Client applications

Application	Technology	Client	Authentication type
Banking application	IBM System z	IBM Personal Communications	Host based
Credit Card Processing	IBM System i	IBM Personal Communications	Host based
E-mail	Lotus Domino	Lotus Notes client	Windows application
Chat	Lotus Sametime	Lotus Sametime Client	Windows application
ERP	SAP	SAP Client	SAP application
Intranet	WebSphere Application Server	Firefox / Internet Explorer	Web based application

Attention: The previous architecture only referenced the components that are important for this scenario. This is not a complete architecture as there are a lot of components missing. Components like networking, firewalls, and so on, were left out for clarity purposes.

16.3 Current password management problems

The current applications and directory infrastructure deployed by Areally Big Investment Corp. present some problems mainly because there is a mix of different applications, operating systems, and directories, and most of the applications use their own user repository. Because of this, for every application, the user has to maintain a separate username and password, which must adhere to the application's password constraints that typically include one or more of the following:

- Minimum number of characters
- Must contain a specific number of uppercase or lowercase characters
- Must contain at least one number
- Must contain at least one special character
- Cannot repeat any of the previous number of passwords used for that application

Because of those constraints and other external factors, a lot of problems have started to appear.

16.3.1 Time and money related problems

Users of the company have started to complain and become frustrated because of the number and complexity of passwords they need to memorize. This is forcing the users to write down the passwords either on paper notes that they keep close to their desktops or in unencrypted text files on their computers.

In the recent past the total number of password related calls to the help desk increased to about 30% of all calls received. Furthermore, the procedure for a password reset takes time to be completed, time that users cannot log into their desktop or application.

This in itself creates a bigger problem in terms of auditing. If a user has to complete a specific task, and the password for the application to execute this task was forgotten, the user will probably ask his coworker to log in with his own credentials instead of the actual user's credential (which he does not remember). This action, considered harmless for most users, creates a missing audit trail, because all tasks done by the user are recorded under the other user's credentials; therefore, making him implicitly responsible for the actions taken.

The procedure for every password reset call also has a price. A framework needs to be in place to reset the password from the help desk, proper access to the administrative consoles of the applications or operating systems, and so on. Also, added to these, is the cost of having a user without being able to access his computer or application for the time that it takes for this procedure to be

completed. Analyst groups estimated this cost to be between US $15 and US $45 per call.

16.3.2 Security related problems

From a business perspective some applications might require enhanced access control mechanisms, such as biometric authentication, additional password constraints, and so on. For some existing or older applications, it can be too complicated or risky to modify them in order to handle the new security requirements. Because of the risk of modifying these applications, the IT security department must find a different way to satisfy the requirements without compromising the applications.

16.3.3 Compliance with regulations

To comply with regulations is no longer a choice for a company. The need to audit password related user activities becomes more important every day. To be able to record the applications into which a user has logged in, when a password was changed, or to assure the right security controls mechanisms becomes more important by the day.

16.3.4 Current single sign-on costs

After studying the current password related problems the security management team at Areally Big Investment Corp. determined that the company is incurring a high amount of unnecessary costs due to their current problems.

To mitigate these costs, the CIO's team took into account the following variables:

- Number of users to implement the solution: 5000
- Number of applications to implement single sign-on: Seven
- Cost of one password reset call: Between US $15 and US $45[1]
- Number of times a user forgets a password per application per year: Two
- Total solution costs the first year: $425,000
  - Solution licensing costs: $400,000
  - Implementation services costs: $25,000

[1] Various analysts firms have estimated this number between US $15 and US $45 per call. The actual costs depend on different factors like the process, hourly wage of employees, and so on.

Important: To calculate licensing costs, we assume US $80 per user costs for the Tivoli Access Manager for Enterprise Single Sign-On core, the Authentication Adapter, and the Desktop Password Reset Adapter. This is a reference price for this particular example. For actual licensing costs of this product, call your local IBM sales representative.

The implementation services costs are assumed typical for a project this size, but they do not represent a real services offering from IBM or any IBM business partner and therefore hold no validity as an offer.

The following assumptions and calculations were made to get to the yearly costs:

- Number of times users requests a password change from the help desk: 14[2].
- Costs of those calls per user per year: 14 calls x $15 per call: $210
- Costs for the 5000 users: 5000 x $210 = $1,050,000

Based on the fact that password reset calls cost the company $1,050,000 per year, and an enterprise single sign-on solution implementation costs $425,000, the return on investment (ROI) is 247% ($1,050,000 / $425,000).

If we modified the scenario above to increase the costs of every call to $45 per call, the results are the following:

- Number of times users requests a password change from the help desk: 14.
- Costs of those calls per user per year: 14 calls x $45 per call: $630
- Costs for the 5000 users: 5000 x $630 = $3,150,000
- Return on investment: 741% ($3,150,000 / $425,000).

Based on the above calculations, the security team reached the conclusion that this solution not only has a very high return on investment rate, but can save the company a lot of money for every consecutive year by reducing password reset calls to the help desk.

[2] This is assuming that every user will request 2 password changes per application per year. We know that not every user requests it, but because some user request passwords more times than the established, and some users less times, this number works for the exercise.

16.4 Business requirements

Let us take a look at the business requirements, as defined by the CIO office of the corporation:

- To increase the employee's productivity and reduce costs, a password management solution must be implemented.
- The solution should be operated efficiently and correctly in order to comply with the corporate security policies.
- To lessen administrative cost, it is desirable to automate management operations related to passwords wherever possible.
- In order to reduce the probability of fraud, strong authentication has to be evaluated for specific applications.
- The new system has to integrate well into the existing infrastructure without making significant modifications and investments to it—making full use of the existing resources.
- The new system has to allow the users to also use the corporate system for their personal credentials.

16.5 Functional requirements

Based on the corporate business vision and business needs, Areally Big Investment Corporation came up with the following list of functional requirements that have to be met by the proposed solution.

- Externalized authentication

  The proposed solution must provide externalized authentication and authorization at an enterprise level including Web based applications, client server based applications, and distributed platforms.

- Integration with current infrastructure

  The proposed solution must make use of the current directory by leveraging the directory services already in place.

- Support for the following application types:
  - Web Applications running in Internet Explorer of Firefox Web browser.
  - SAP client application single sign-on.
  - Existing applications have to be accessible by integrating SSO to host/mainframe based applications that use an emulator based client application.

- Support for offline mode

  The proposed solution must be able to cache the credentials on a user's workstation, so it works in a disconnected environment.

- Integration for non-enterprise applications

  The solution must allow users to integrate non-enterprise applications into their personal SSO applications without any need for programming or scripting.

- Support for automatic password change without user intervention

  The proposed solution must support the changing of a password when the target application requests it, generate a new password, and update it to the central repository without any user intervention.

- Support for strong authentication

  The solution must support the use of different strong authentication mechanisms. It must especially support biometric devices.

- Provide self-service capabilities for the single user password

  The proposed solution must offer capabilities to the user in case they forget the password. There has to be a way to choose a new password in a secure way without having to call the Help Desk.

- Provide auditing capabilities

  The proposed solution must log password related events, such as credential changes, log into an application, agent shutdown, and so on.

- Enforce encryption for all data so no credentials are stored in plain text

  The proposed solution must encrypt data stored in the server, in the client agent, and in transit. The encryption method chosen must be certified to meet FIPS 140-2[3]. It must encrypt all communications with the SSO server using SSL for all communications with the respected credential repository.

16.6 Design approach

Now that the functional pre-requisites are defined, let us consider a more detailed description of the architecture and how Areally Big Investment Corp. is going to implement IBM Tivoli Access Manager for Enterprise Single Sign-On to satisfy all of the requirements described in 16.5, "Functional requirements" on page 498.

[3] For more information on FIPS 140-2 please visit http://csrc.nist.gov/cryptval/140-2.htm

The products and components that will be deployed include:

- IBM Tivoli Access Manager for Enterprise Single Sign-On core component
- IBM Tivoli Access Manager for Enterprise Single Sign-On Desktop Password Reset Adapter
- IBM Tivoli Access Manager for Enterprise Single Sign-On Authentication Adapter

The following is a diagram (Figure 16-2) that shows the initial deployment of the core solution component:

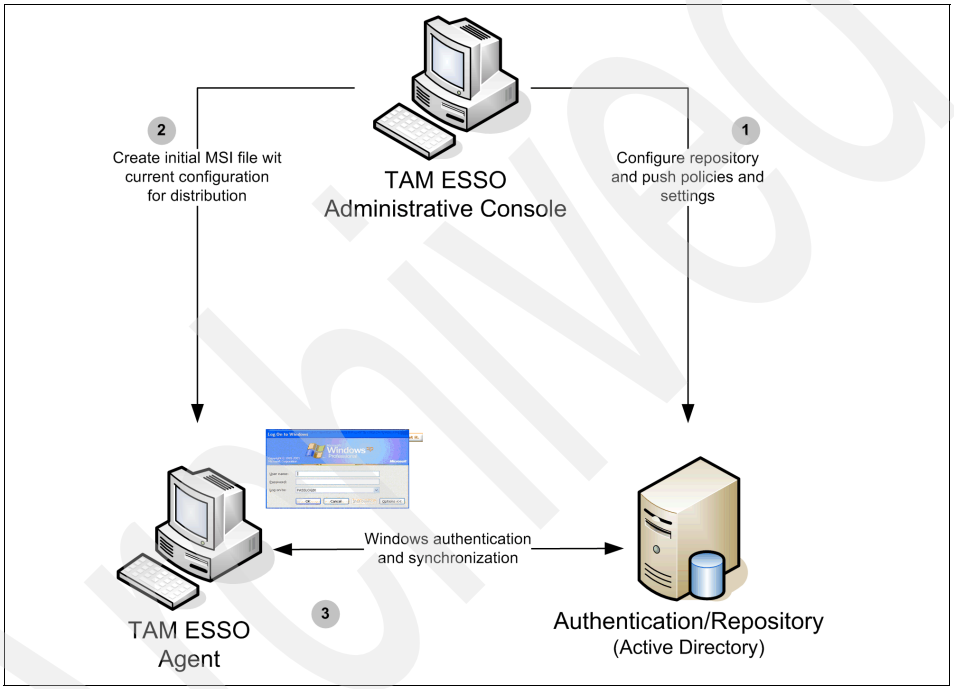


Figure 16-2 Core solution deployment

16.6.1 Core solution deployment

The first part of this deployment is to set up the administration console to configure the repository, which in our case is the Microsoft Active Directory. All credentials, settings, and configuration information is stored in the repository.

Important: The following parts describe some details of a deployment of Tivoli Access Manager for Enterprise Single Sign-On for this architecture, but it is not considered a complete deployment guide. For more information on a deployment guide for this solution, please visit the following Web site:
http://www.redbooks.ibm.com/redpieces/abstracts/sg247350.html

Currently, the templates for the different applications must be created and configured. For every application the following must be done:

1. Configure the initial logon form.
   - If it is a Windows application, it occurs based on the specific fields in the logon dialogs.
   - If it is a Web based application, it is based on the authentication form.
   - If it is a host/mainframe application, it is based on the coordinates of the login and password fields in the emulator window.
2. Configure the password change form based on the previous procedure.
3. Test the template.
4. Deploy the template to the repository.

After the configuration for all required applications occurs, a custom MSI file must be created in order to deploy these settings to all the workstations.

After the installation is completed for all the workstations, the agents can synchronize with the Active Directory to retrieve templates and settings. At this point, users can add their credentials and start experiencing single sign-on.

16.6.2 Desktop Password Reset Adapter deployment

The next step in our solution is to deploy the Desktop Password Reset Adapter (DPRA). The following is a diagram (Figure 16-3 on page 502) of the architecture of this component.

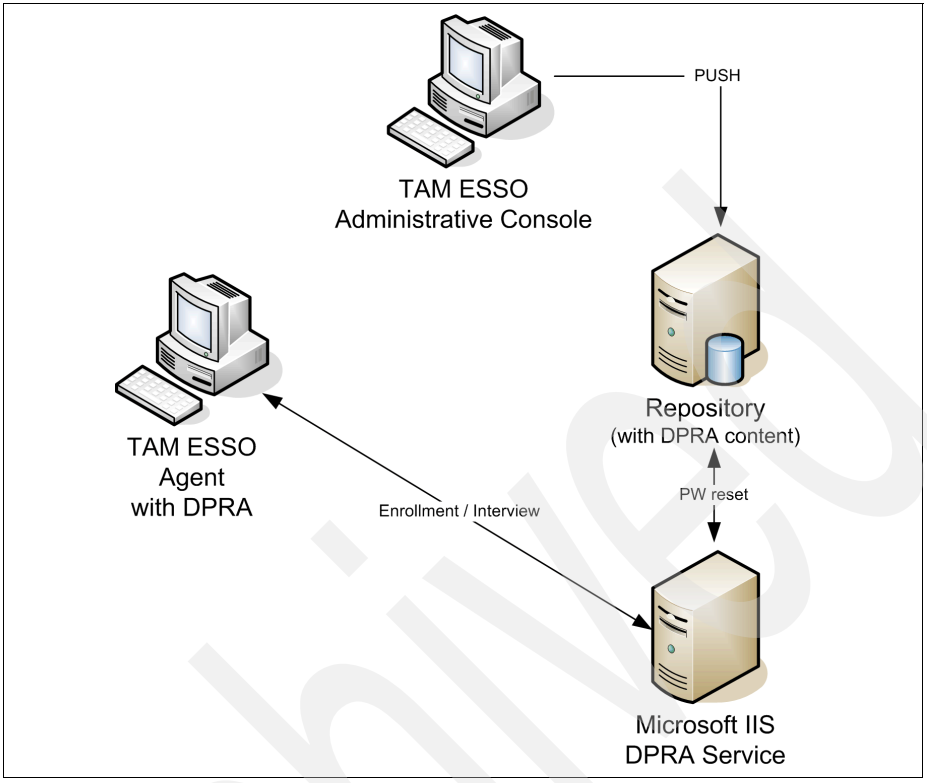


Figure 16-3 Desktop Password Reset Adapter deployment

The DPRA component allows the users to reset their only password (the domain password) directly from their locked workstation. Before having the service available for the users, they must answer a few predefined questions. In the case where a user forgot the primary password, these questions authenticate the users and give them the chance to change their password.

From an architectural point of view the DPRA has two main components, the server and the client. It uses the Microsoft Internet Information Services to host the Web application that conducts the interview and the actual reset. For more information on the architecture and functionality of this component see 15.1.8, “Desktop Password Reset Adapter” on page 467.

In our solution, the server component is deployed and the client is distributed to all the workstations. An important part of this deployment step is to educate the users on the procedure to sign up for the service and how to use the password reset activities. This way, all of the users have to sign up, answer the questions, and start using this service when they forget their password.

16.6.3 Authentication Adapter deployment

In our solution, there is a requirement to support strong authentication for some specific applications. To satisfy this requirement, we use the *graded authentication* functionality of the Authentication Adapter, which allows us to require an additional level of authentication for a specific application. This way, the enterprise can selectively deploy strong authentication devices, like smart cards or biometric devices, for a group of users. No changes are required for applications. The strong access control enforcement only affects the workstations.

Tip: For more information on the installation and configuration of the Authentication Adapter visit the Tivoli Access Manager for Enterprise Single Sign-On Information Center at the following Web site:
http://publib.boulder.ibm.com/tividd/td/IBMTivoliAccessManagerforEnterpriseSingleSign-On6.0.html

16.7 Solution analysis

After a description of how the different components fit in the architecture for Areally Big Investment Corporation, we now analyze the different requirements and see how our solution answers these requirements.

- Externalized authentication

  Tivoli Access Manager for Enterprise Single Sign-On provides single sign-on by introducing a secure middle layer that authenticates the user once and detects all subsequent requests for credentials from all applications, including Web, client/server, and host/mainframe based applications.

- Integration with current infrastructure

  Tivoli Access Manager for Enterprise Single Sign-On makes use of the current Active Directory in place in the corporation. It also supports other LDAP compliant directories or databases if required.

- Support for the following application types:
  - Web applications running in Internet Explorer or Firefox Web browser: Tivoli Access Manager for Enterprise Single Sign-On supports the sign on to a Web application either in a form or a pop-up window. It looks for the correct fields and injects the credentials accordingly.
  - SAP application: Tivoli Access Manager for Enterprise Single Sign-On supports logging into the SAP client.

  - Existing applications based on host/mainframe emulators: Tivoli Access Manager for Enterprise Single Sign-On responds to credential requests from host/mainframe applications by interacting through the terminal emulator used to access the application. The agent monitors the window presented in the emulator and looks for specific text to identify the correct screen and to inject the credentials in the correct place.

- Support for offline mode

  Tivoli Access Manager for Enterprise Single Sign-On supports working in offline mode by keeping a local copy of the credentials on the local workstation. When the user connects to the network, the credentials are synchronized with the repository.

- Integration for non-enterprise applications

  Tivoli Access Manager for Enterprise Single Sign-On recognizes login requests from non enterprise Web applications, for example a Web based e-mail application, or an online banking application. If allowed by the administrator, the user can also add these credentials to the repository to utilize single sign-on for these applications.

- Support for automatic password change without user intervention

  Tivoli Access Manager for Enterprise Single Sign-On can be configured to automate the password change process for the user. In this case, when a password change request occurs, the agent generates a new random password according to the policy, makes the change in the application, and stores this new password in the credentials repository for later use.

- Support for strong authentication

  Tivoli Access Manager for Enterprise Single Sign-On supports various authentication mechanisms including smart cards, biometric devices, tokens, and digital certificates at the workstation.

- Provide self-service capabilities for the single user password

  In case the users forget their password Tivoli Access Manager for Enterprise Single Sign-On offers them the capability to reset the password by choosing another one in a secure way without having to call the Help Desk. Authentication to the password reset mechanism is enforced by using a user pre-defined question algorithm.

- Provide auditing capabilities

  Tivoli Access Manager for Enterprise Single Sign-On logs all user password events, including last successful logon, number of login attempts, credential changes, and agent events. It also logs dates, times, and usernames for events such as application log ons, application password changes, startup and shutdown of the agent, and setting changes.

- Enforce encryption for all data so no credentials are stored in plain text

  All local credentials and those stored in the repository are encrypted using Triple DES symmetric key encryption, certified to meet FIPS 140-2. If desired, the encryption algorithm can be changed. Tivoli Access Manager for Enterprise Single Sign-On utilizes SSO component communication by default.

16.8 Conclusion

Like many other large companies, Areally Big Investment Corporation battled the password management problem for years, suffering productivity loss, elevated user help desk costs, and became more vulnerable due to this fact. The implementation of an enterprise single sign-on solution is the main step taken to mitigate these issues.

The way the company is approaching the security problem covers the following important issues:

- Improving the user experience around accessing different applications
- Enforcing general security policies of password creation
- Reducing operational costs
- Complying with regulatory standards

Finally, this chapter described how the different components of IBM Tivoli Access Manager for Enterprise Single Sign-on address different authentication requirements for a company with characteristics such as mobile users, strong authentication, and a diverse variety of applications.

If you want to learn more about deploying a Tivoli Access Manager for Enterprise Single Sign-On solution refer to the *Deployment Guide Series: Tivoli Access Manager for Enterprise Single Sign-On*, SG24-7350.

Part 3

Managing identities and credentials

In Part 3, we discuss the solutions Tivoli offers in the identity and credential management space of the overall security architecture. Identity and credential information, which generally revolves around managing individuals, can be handled by Tivoli Identity Manager and Tivoli Directory Integrator. These products handle a multitude of integration aspects with all sorts of IT infrastructures and application environments, which are detailed throughout this part.

17

Identity management

In this chapter we introduce another discipline in our enterprise security architecture saga: identity or user lifecycle management.

Identity management is a comprehensive, process-oriented, and policy-driven security approach that helps organizations consolidate identity data and automate the deployment across the enterprise. In this chapter we attempt to outline methods of identifying the key components of an identity management architecture using IBM Tivoli Identity Manager.

17.1 Business drivers

In order to effectively compete in today's business environment, companies are increasing the number of users (customers, employees, partners, and suppliers) who are allowed to access information. As IT is challenged to do more with fewer resources, managing user identities and their access to resources throughout the identity lifecycle is even more difficult. Typical IT environments have many local administrators using manual processes to implement user changes across multiple systems and applications. As identity management grows more costly, it can inhibit the development and deployment of new business initiatives.

An integrated identity management solution can help get users, systems, and applications online and productive fast, and maintain dynamic compliance to increase the resiliency and security of the IT environment, while helping to reduce costs and maximize return on investment. An identity management solution has three key areas:

- Identity lifecycle management (user self-care, enrollment, and provisioning)
- Identity control (access and privacy control, single sign-on, and auditing)
- Identity foundation (directory, directory integration, and workflow)

As the world of on demand gains global acceptance, the traditional processes of corporate user administration are no longer able to cope with the demands of increased scale and scope expected from them. Identity management is a super-set of older user provisioning systems that allows for the management of identity and credential information for customers, partners, suppliers, automated processes, corporate users, and others.

As organizations come to depend on their IT assets more, these assets attract the attention of accounting and reporting standards. IT data and system assets will increasingly become balance sheet line items, and therefore be subject to different audit and compliance rules. Organizations must be able to demonstrate due care, due diligence, improved security, and compliance with other financial rules. We should realize that any entity using the IT systems run by an organization must be included in the scope of identity management if we are to have any chance of achieving these goals.

17.2 Issues affecting identity management solutions

Undertaking an identity management project reveals situations that are not always readily apparent. Two major areas of interest: enabling user access (session management, authorization, authentication, and so on) and user lifecycle management (user administration, provisioning, and so on) stand at the

forefront. Each area has many facets of its own. Tivoli Identity Manager is primarily concerned with user lifecycle management, and Tivoli Access Manager is primarily concerned with session management.

Identity management takes a lifecycle approach to the management of an identity and access control from the beginning of the process. Specifically, identity management handles the changes that occur during the lifetime of the user's account. This specifically means that it has the ability to integrate with pre-existing information sources within the enterprise, such as directories and HR systems. This gives a complete approach to identity management by leveraging the existing information in data directories as well as integrating and utilizing the HR system to access information about an employee (hirings, promotions, transfers, termination of employment). In this manner, the identity is managed through all stages of the process to ensure consistent handling of the identity. This holistic lifecycle approach helps to minimize the risk to the enterprise because it is ordered rather than fragmented.

When managing identities and access control, an integrated approach should be taken to ensure the integrity of the company and the protection of its assets. Integration is the key to effectively managing an individual identity and access. An easy example of the possible risk that could be encountered if the identity management was not integrated is when updating the HR database to reflect the termination of an employee. Because there is a lag time between the HR department notifying the various systems administrators, the various systems that the former employee had accounts on (company intranet, extranet) are still active. This means that, effectively, the former employee has access to sensitive company data. If this employee has taken a position with a competitor while still having access to the data, this leaves the former employer open to risk.

Clearly, many obstacles exist but there are best practices that organizations can follow to mitigate risk, optimize investment, achieve results, and ultimately balance user experience with greater productivity and cost savings, allied to increased IT security.

17.3 Security policies, risk, due care, and due diligence

The senior management team of an organization has to show due care in all dealings, including security-related matters. Showing due care helps to create a professionally managed organization, which in turn helps maintain shareholder value. Due care can also be an important step toward avoiding claims of negligence. From a security perspective, showing due care can be achieved by having well-thought-out security policies.

Security policies have to balance a number of conflicting interests. It is easy to write security policies that deny access or make access controls so onerous that either no business gain can be achieved or the security policies are ignored. Security policies must set a sensible level of control that takes into account both the culture and experience of the organization and an appreciation of the risks involved.

Risk assessment is an important topic in its own right but is outside the scope of this chapter. Briefly, risk is usually assessed either formally or informally using quantitative or qualitative methods. This can be as structured as a full external risk assessment, or simply based on the intuition of members of an organization who know and understand how their business is constructed and the risks involved. Risk can be dealt within any of four ways:

Transfer The most common way of transferring risk is through insurance. In the current economic environment, the availability and cost of insurance is variable. Currently, this method is more volatile than in the past.

Mitigate Mitigation of risk can be achieved by identifying and implementing the means to reduce the exposure to risk. This includes the deployment of technologies that improve the security cover within an organization. Deploying an identity management tool mitigates the security risks associated with poor identity management.

Accept An organization may chose to accept that the impact of the risk is bearable without transferring or mitigating the risk. This is often done where the risk or its impact is small, or when the cost of mitigation is large.

Ignore Often confused with risk acceptance, ignoring risk is all too common. The main difference between accepting risk and ignoring risk is that assessment is an implicit part of risk acceptance. If no valid risk assessment has been done, this should raise a warning flag that points toward the dangerous path of ignoring risk.

Understanding the risks that exist enables us to write appropriate security policies. Having security policies shows the exercise of due care, but unless the policies are implemented, due diligence cannot be shown. Many organizations write good security policies only to fail at the implementation stage, because implementation represents a difficult or costly challenge. In the next section, we show how a centralized identity management solution can be used to enforce security policies relating to identity management. This gives us demonstrable due diligence with respect to identity management.

17.4 Centralized user management

Identity management is the process of managing the information for a user's interaction with an organization. As such, it is an important element of e-business security and is vital to sustaining a healthy e-business. Without a solid identity management solution, problems can occur when users—whether they are employees, customers, business partners, or suppliers—require access to IT resources. The benefits of centralizing the control over user management, while still allowing for decentralized administration, affects two key business areas: the cost of user management can be reduced and security policies can be enforced.

The capabilities of an identity management solution can be classified into eight levels. These capability levels can readily be arranged into a pyramid as shown in Figure 17-1, the base of which is the most core required capability of the provisioning solution. After the capabilities in the lowest level are addressed, you can move up to the next level in the pyramid, which provides increasingly more powerful capabilities. The ideal provisioning solution addresses all eight levels.

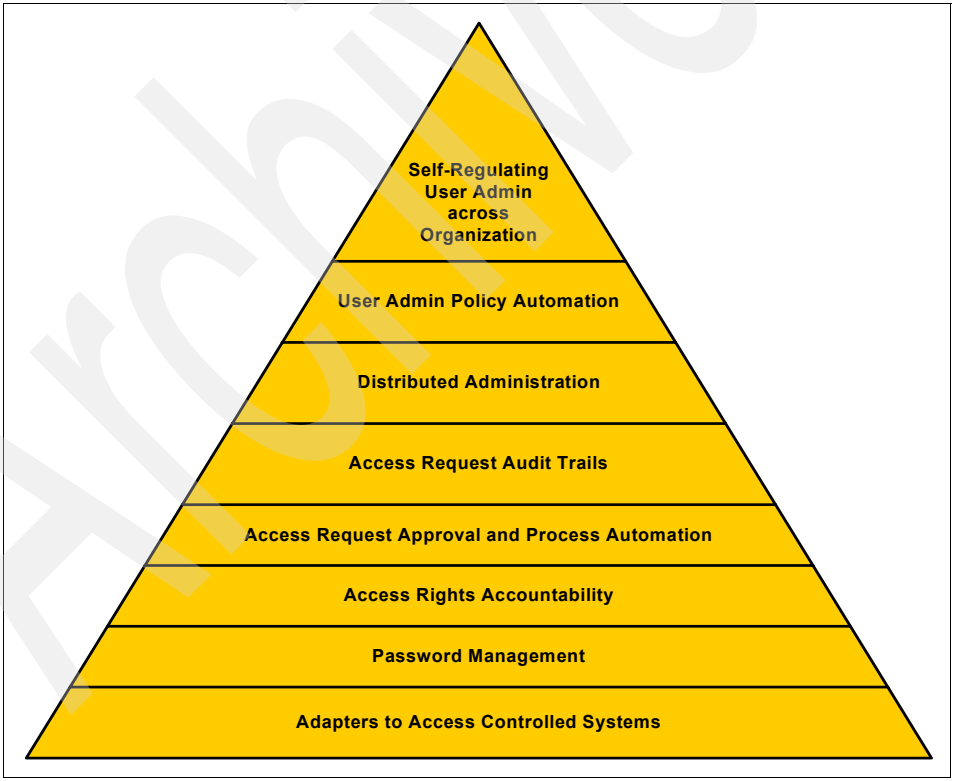


Figure 17-1 The eight levels of identity management

17.4.1 Adapters to access controlled systems

In order to automate provisioning, the solution must communicate securely with each target system being managed. If this adapter does not exist, then an administrator must still make the required changes manually. The adapters are the key mechanisms that translate the commands of the provisioning solution into the proprietary language understood by the managed resources. Further, the richness of the language used is important. Managed systems (SAP, for example) support the definition of hundreds of parameters describing user access. The adapter must support the needs of the managed system and the needs of the organization in creating or changing accounts.

Communication between the provisioning solution and the managed system must be bidirectional, secure, and bandwidth-efficient. Bidirectionality is critical to capturing changes made directly to the managed system and reporting the change to the provisioning solution for evaluation and response. The link must be encrypted so that no one can listen in and steal authentication information such as passwords. The link must also allow authentication of the source so that a new command cannot be injected into the system by an imposter to create an inappropriate account.

Last, because the managed resources are physically distributed across the corporate wide area network (WAN) or the Internet, bandwidth efficiency must be considered. These networks often have limited available capacity and are expensive, requiring the provisioning solution to operate with as little overhead as possible.

When we talk about managed systems, we have to look at two types of repositories:

- User repositories
- Endpoint repositories

User repositories

User repositories contain data about people, and most companies have many user repositories and will continue to add new ones due to new and custom applications. These can be:

- Human Resources systems
- Applications
- LDAP and other directories
- Metadirectories

Endpoint repositories

Endpoint repositories contain data about privileges and accounts, and most companies have a great variety of these repositories implemented throughout their environment. Some of these are:

- Operating systems, such as Linux, Windows XP and AIX
- Tivoli Access Manager
- Tivoli Federated Identity Manager
- Network devices
- RACF
- Lotus Notes
- ERPs
- Databases

Therefore, it is important when considering centralized identity management systems to be sure that the coverage of the system takes both types of repositories into full account. These repositories hold a wealth of identity-related information. Tying all of the information together rather than duplicating it is cost-effective and eliminates mistakes.

17.4.2 Password management

Password management is the ability to control password quality and change passwords throughout an environment. As companies deploy more and more systems that contain access controls, the number of passwords required to be remembered by each user increases. This increase poses a risk to the organization as more users have a tendency to write down their passwords in order to keep track of them. A costly side effect of this is the increased workload on the help desk to reset forgotten passwords. (Research shows that approximately 30% of total calls to the average help desk are for password-reset assistance.)

Password strength is also problematic for many organizations. Hackers possess effective tools and techniques for cracking poorly constructed passwords. Organizations desire to enforce stronger password formation rules across the enterprise but must balance that desire against poor end-user experience and increases in forgotten passwords.

Password management capabilities enable users to self-service their own accounts. Users visit a Web-based system, authorize themselves, then may reset or synchronize their passwords on all of their accounts. Further, the passwords they select can be evaluated against rules on their formation to ensure uniform conformance with organizational password policies. A user typically has multiple accounts and passwords. The ability to synchronize passwords across platforms and applications provides ease of use for the user. It

can also improve the security of the environment because each user does not have to remember multiple passwords and is therefore less likely to write them down.

Key points to password management include:

- User self-service through the Web without logging onto the network
- Challenge-response system to authenticate a user with a forgotten password by using shared secrets
- Ability to implement password formation rules to enforce password strength across the organization
- Ability to synchronize passwords for multiple systems to the same value to reduce the number of different passwords to be remembered by the user
- Delivery of password-change status (success or failure) to requestor
- Ability to securely deliver passwords to users for new accounts

17.4.3 Access rights accountability

Tracking precisely who has access to what information across an organization is a critical function of the provisioning solution. Not only does it allow control of sensitive systems but it should expose all accounts that have unapproved authorizations or authorizations that are no longer necessary. These inappropriate accounts pose one of the most serious threats to corporate security because they are valid, active accounts so they cannot be detected as a traditional cyber-attack. Access rights accountability provides configuration control over all accounts and their specific authorities.

Orphan accounts are those active accounts found on many systems that cannot be associated with a valid user. Improperly configured accounts are those associated with valid users but granted improper authorities. These accounts may appear at any time due to local administrators retaining rights to use local administrative consoles. In enterprise-wide environments, these local consoles cannot be disabled because of their multiple operational use. The key to the control of improper and orphan accounts is, on a continuous basis, to associate every account with a valid user and maintain a system-of-record detailing the approved authorities of the account. When the user's status with the organization changes, their access rights must change too. If the account configuration changes, it must be compared with an approved configuration and policy.

The ability to control orphan accounts requires that the provisioning system link gather account information with authoritative information about the users themselves. Authoritative user identity information is typically found in Human

Resources and various databases and directories containing information about users in other businesses.

The ability to control improper accounts is much more difficult. It requires a comparison of the desired with reality at the account-authority level. Simple existence of an account does not expose its capabilities. Accounts in sophisticated IT systems include hundreds of parameters defining the authorities; these are the details that must be controlled.

Accounts found to be orphaned or improperly configured must be reported and corrected. Provisioning solutions should notify the proper personnel to fix account settings.

Access rights should include:

- Flexible mechanisms to connect to multiple data stores containing accurate information about valid users
- Ability to load identity store information about a scheduled or event basis
- Ability to detect and respond to identity store changes in near-real time
- Ability to retrieve account information from target managed resources on a scheduled basis, both in bulk or in filtered subsets to preserve network bandwidth
- Ability to detect and report in near-real-time local administrator account maintenance (creation, deletion, changes) made directly on local resources
- Ability to compare local administrator changes against a system-of-record of account states to determine whether changes comply with approved authorities and policies
- Ability to notify designated personnel of access-rights changes made outside the provisioning solution
- Ability to compare account user IDs with valid users to identify accounts without owners (orphans)
- Ability to automatically suspend or delete a detected orphan account
- Ability to automatically suspend or roll back a reconfigured account that violates policy
- Ability to examine reports on orphan accounts
- Ability to readily view the accounts associated with a user or a resource
- Ability to assign discovered orphan accounts to a valid user

17.4.4 Access request approval and process automation

Access request approval and process automation is a key component in rapidly and accurately changing user access rights. The approval processes are a specialized form of workflow that determines, based on organizational policy, the need to approve a requested change to access rights prior to its execution. Many organizations still rely on paper and e-mail forwarded in many different paths through the organization.

These approaches can be very slow. Requests can sit idle in an inbox or be rejected because they are missing key information; consequently, the process must begin again. A complete provisioning workflow solution automatically routes requests to the proper approvers and escalates to alternates if action is not taken on the request in a specified time. This workflow automation can turn a process that typically takes a week into one that takes only minutes.

Some organizations also require that information about accounts or background information be added to the request as it flows through the process. This information may come from users involved in the process or it may be computed or extracted from other systems.

A workflow automation tool should offer the following features:

- Web-based mechanism for requesting access to a system
- Automatic approval routing to the persons appropriate to the system access requested and organizational structure
- Review and approval mechanisms that offer a zero-footprint client
- Ability to use defined organizational information to dynamically determine routing of approvals
- Ability to delegate approval authority to another person
- Ability to escalate a request to an alternative approver if the allotted time elapses
- Ability for different personnel to view different levels of information based on their job duties
- Ability to request information from approval participants to define account-specific information during the process
- Ability to determine service instances where a physical account should be created
- Ability for the system to change account information in the managed resources of a specific organization

- Ability to request information from specific participants in the workflow process
- Ability to request information from external functions, applications, and data stores during the process
- Ability to easily create, design, and modify a workflow via a graphical drag-and-drop interface

17.4.5 Access request audit trails

Traditionally, many organizations have treated audit logs as places to look for the cause of a security breach after the fact. Increasingly, this is seen as an inadequate use of the information available to an organization, which would be exhibiting better due diligence by monitoring and reacting to logged breaches in as near to real time as possible.

Centralized audit trails of access requests are an important aspect of supporting independent audits of security practices and procedures in an organization. These audit trails capture all aspects of the administration of access rights, from initial access requests to changes in account details. Security audits are part of every organization, whether they are conducted by internal security audit teams or are external audits supporting formal bookkeeping. If recordkeeping is incomplete, inaccurate, or stored in multiple locations, then these audits can consume extensive time and human effort to conduct. Audits are frequently disruptive to daily work efforts but are mandatory for the safe and secure operation of the organization. Among other things, audit teams look for orphan accounts or inappropriate access privileges that exist on important systems. Audits may occur from once a quarter to as frequently as once a week, depending on the organization.

An access request tool must include the following:

- Time-stamped records of every access change request, approval or denial, justification, and change to a managed resource
- Time-stamped record of every administrative and policy-driven change to access rights
- Time-stamped record of any encountered orphan accounts and bypasses of administrative systems
- Convenient, flexible means of running reports that show audit trails for users, systems, administrators, and time periods
- Audit trail that is maintained in a tamper-proof environment

17.4.6 Distributed administration

Distributed administration enables the administrative tasks involved with provisioning, whether manual or automated, to be distributed securely among various departments, organizations, or partners. This is important for two reasons: accuracy and scale. It is wise to move the process of requesting and approving access changes close to the people who know whether the resource is truly needed by the individual.

Further, this distribution allows the workload to be balanced across a large number of administrators rather than a single dedicated and centralized team. This becomes fundamental in large organizations with multiple regional offices and those with multiple business partners. Distribution should be performed all the way down to the individual level when desired for self-service or self-enrollment. To accomplish this, the system must support delegated administration and user authentication. Delegated administration enables the responsibilities for using and changing identity information to be delegated down through an organization in a controlled manner.

Administrative tasks such as requesting access for a user, approving a change, or defining local policies can be delegated to individuals throughout an organization or its partner network. In this way, individuals that have the most accurate knowledge of users' needs can request or approve changes. Lower levels can define local policies for access rights assignment within the guidelines created by the organization. A key aspect of delegated administration is filtering the information presented to an administrator on a need-to-know basis. Not only does this make the system more usable to the administrator, but it also prevents exposing information to personnel without a need to know. For example, external business partners may be administering their own users into a common supplier environment but each business partner must remain invisible to the other business partners.

Authentication to the system becomes critical at this point. As widely dispersed individuals may make changes affecting access rights for others, it is critical that system security be maintained. Frequently these interfaces are accessible to users over the Internet, and that requires stronger authentication approaches. To meet the need for stronger authentication, the solution should enable you to use your own custom authentication mechanisms.

An administration tool should be able to:

- Define organizational structures based on the access-granting authorities of an organization
- Delegate each administrative task with fine-grained control (for example, approval authority, user creation, workflow definition)

- Delegate administrative tasks to n-levels of depth
- Access all delegated capabilities over the Web with a zero-footprint client
- Create private, filtered views of information about users and available resources
- Incorporate Web access control products to include the provisioning solution within the Web single sign-on environment
- Incorporate custom user authentication approaches commensurate with internal security policies
- Distribute provisioning components securely over WAN and Internet environments, including crossing firewalls

17.4.7 User administration policy automation

User administration policy automation is the way to evaluate and enforce business processes and rules for granting access. Role Based Access Control (RBAC) is a method of granting access rights to users based on their assignment to a defined role in the organization. Provisioning solutions that embody RBAC or other types of rules that assign access rights to users based on certain conditions and user characteristics are examples of user administration policy automation.

Automation is key to managing large numbers of users across disparate resources and assigning, monitoring, and revoking user entitlements. The solution should enable users to be defined as members of groups, including roles. Entitlements to resources for these groups of users are defined in the security policies. Any change to information about a user should be evaluated to determine whether it alters the user's membership to a group. If there is an effect, policies must be reviewed and changes to entitlements must be put into place immediately. Likewise, a change in the definition of the set of resources in a policy may also trigger a change in entitlements.

The following elements should be included in user administration policy automation:

- Ability to associate access-rights definition with a role within the organization
- Ability to assign users to one or more roles
- Ability to implicitly define subsets of access to be unavailable to a role
- Ability to explicitly assign users individual access rights
- Ability to dynamically and automatically change access rights based on changes in user roles

- Ability to define implicit access rights available to users in a role upon their request and approval
- Ability to use defined organizational information to dynamically determine routing of approvals
- Ability to detect, evaluate, and respond to user authority changes made directly to a managed resource
- Ability to report on roles, rights associated with roles, and users associated with roles
- Ability to set designated times for changes in access rights or policies
- Ability to create unique user IDs consistent with policies and not in current use or previous use by the organization
- Ability to create user authorizations extending an existing account
- Support for mandatory and optional entitlements (optional entitlements are not automatically provisioned but may be requested by a user in the group)
- Support for entitlement defaults and constraints (each characteristic of an entitlement may be set to a default value, or its range can be constrained, depending on the capabilities of the entitlement to be granted)
- Ability to create a single account with multiple authorities governed by different policies
- Ability to create user IDs using a set of consistent algorithms defined by the organization

17.4.8 Self-regulating user administration across organizations

This ultimate level of the hierarchy is the ability to provision across multiple organizations that each contain user groups and shared services. In this environment, a change in a user's status is automatically reflected in the access rights inside the user's organization and also in the outside services offered by other organizations. As the provider of services to other organizations, user access rights are automatically established based on your security policies and the assertion of the users authenticity provided by the sponsor or a third party.

Key points of self-regulation include:

- Adherence to open standards
- Secure environment for transmitting access changes across the Internet
- Protection of private user information through secure facilities and sound processes
- Auditing access rights changes

17.5 Lifecycle management

Identity management in general is the process of managing persons and their accounts across all systems. The notion of *lifecycle management* introduces the following concepts:

- The person exists as a *person entity* to the identity management solution. From the time of its creation to its deletion, it will change over time due to external events such as transfers, promotions, leaves of absences, temporary assignments or any other identity-related business process.
- A person who uses an IT asset is considered an account from the identity management perspective. The identity management solution sees this accounts as an *account entity* owned by the *person entity*. The person entity changes will affect its own accounts from the time they are created until the time they are deleted. The person entitlements for each account owned are verified every time the person or account entitlement definitions are changed. There may also be a need to routinely verify that the account is compliant with security policies.

A *lifecycle* is a term to describe how *persons* or *accounts for a person* are created, managed, and terminated based on certain events or a time-based paradigm.

Figure 17-2 on page 524 represents a closed-loop process where a person is registered to use an IT asset, an account is created, and access provisioning occurs to give this person account access to system resources. Over time modifications occur where access to some resources is granted while access to other resources may be revoked. The cycle ends when the person separates from the business and the termination process removes access to resources, suspends all accounts, and eventually deletes the accounts and the person from the systems.

Provisioning solutions are the link between the classical central management solution and the target resources. The capability to quickly negotiate provisioning requirements that map to the identity models and processes of a business is crucial when architecting a solution. The provisioning aspect garners much of the focus and attention. User provisioning is where the process begins, and if provisioning is sluggish or incomplete, users (employees, consultants, customers) develop negative first impressions of the organization.

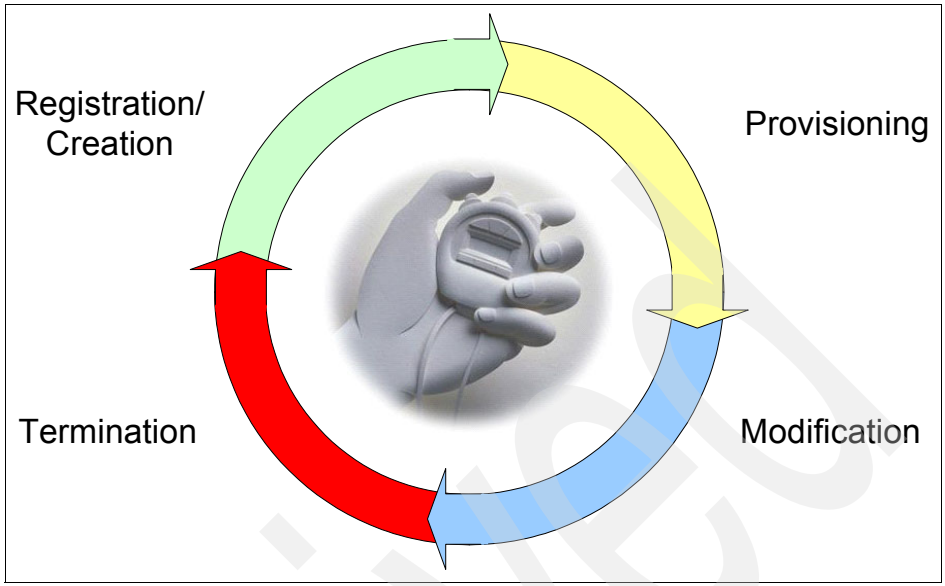


Figure 17-2 User lifecycle management phases

17.5.1 The creation cycle

The creation cycle includes the following:

Person creation The person entity is created with the identity management solution. In most cases person attributes, such as user name, e-mail address, phone number, and other identity-related data, are imported from a person-authoritative system such as a Human Resources (HR) system for employees, a contract system for business partner persons, and other data sources for customer persons.

Account creation The account entity is created on the managed platforms using attributes from the person entity.

17.5.2 The provisioning cycle

The provisioning cycle includes the following:

- Identifying the sponsor (for example, sales or HR), determining the nature of the relationship (customer, internal employee), verifying the user's identity, and assigning a role or roles.
- Fulfillment, which entails gaining approval for the appropriate systems, creating the user's identity in the appropriate directories and repositories, and granting access to those accounts.

17.5.3 The modification cycle

During the maintenance phase of the lifecycle, administrators maintain the following elements:

Person	The person's attributes, such as name, e-mail address, and phone number.
Identity	The user's credentials, such as user name and password, as well as information about the user that may be based on person entity, including name, e-mail address, and phone number.
Access rights	The systems, accounts, and applications the user has access to and the level of access.
Policy management	Updating of access rights based on membership in a particular group or department and consistent enforcement of corporate policies.
Privacy	Enactment of regulations that require enterprises to secure the privacy of certain types of information that are related to specific individuals.

Ideally, users should experience changes in access rights as the organization changes and as their roles within the organization change. The maintenance phase of the lifecycle offers significant opportunity for automation and efficiency gains.

17.5.4 The termination cycle

Termination is the phase with which, from a security perspective, organizations struggle the most. Auditors discovering hundreds or thousands of user accounts that should have been disabled or deleted is common.

During the termination phase, organizations should verify that the relationship between the user and the organization is, in fact, dissolving and disable access

accordingly. Often, accounts are disabled for a term and then deleted. Unfortunately, although this sounds simple, it demands process rigor.

17.5.5 Lifecycle rules

Lifecycle rules provide administrators with the ability to define lifecycle operations (automated processes) to be executed as the result of an event. Lifecycle rules are especially useful in automating recurring administrative tasks. For example:

1. Password policy compliance checking.
2. Notifying users to change their password before it expires.
3. Identifying lifecycle changes such as accounts that are inactive for more than 30 consecutive days.
4. Identifying new accounts that were not used more than ten days following their creation.
5. Notifying users to recertify their account's access to a restricted resource before it is revoked.
6. Identifying accounts that are candidates for deletion because they were suspended for more than 30 days.
7. When a contract expires, identifying all accounts belonging to a business partner or contractor's employees and revoking their access rights.

Table 17-1 describes some lifecycle rules in more detail.

Table 17-1 Sample lifecycle rules

Event	Lifecycle rule	Lifecycle operation
Daily at 12:01 AM	Password expiration	Search all account entities for the Identity Manager and the Access Manager services and generate an e-mail for all user accounts where the password will expire within the next seven days. Where the password is more than 45 days old, suspend the account.
Contract expires	Suspend contractor accounts	Search for all accounts defined for a specific contractor and suspend them at the close of business on the day the contract expires.
Monthly on the first day at 01:01 AM	Recertify Linux account holder	Search all accounts for the Linux service, identify all accounts and send an e-mail to the account holder asking them to recertify their need to use the system.

17.6 Access control models

In 2.6, “Access control models” on page 40 we introduced some of the access control models that are commonly found or are planned for use with a centralized identity management solution—the Role Based Access Control (RBAC), Discretionary Access Control (DAC), and Mandatory Access Control (MAC).

Note: There are many resources available that address access control models. For our discussion we refer to the *CISSP All-in-One Exam Guide* by Harris. Another source you might want to check out is the National Institute of Standards and Technologies at the following Web site:

http://www.nist.gov/

17.6.1 Selection process

The following questions and comments are some of the thought processes that can help choose an access control model and centralized identity management system. Figure 17-3 on page 528 and the questions following it should show the path through the maze. Local, particularly non-functional requirements, may modify the approach you need to take.

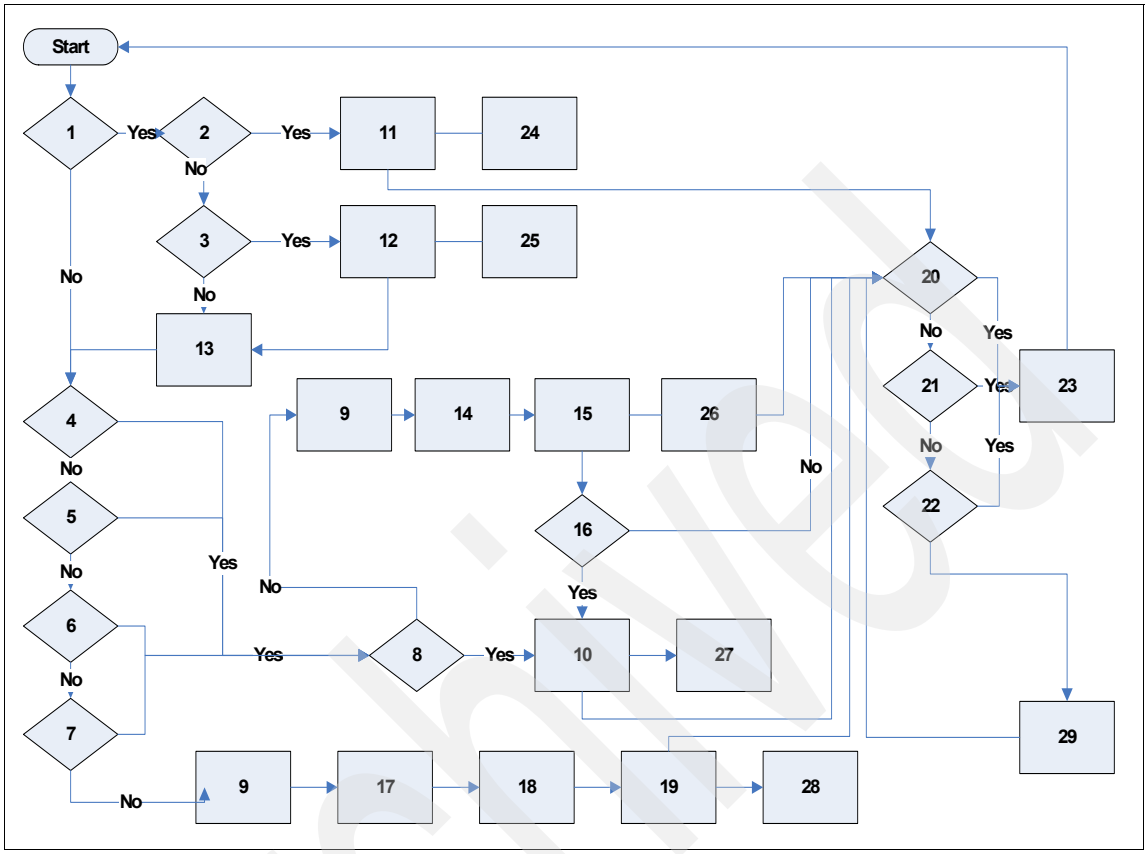


Figure 17-3 Selection flow diagram

Key questions and comments:

1. Does your organization mandate the use of sensitivity silos (confidential, secret, top secret, and so on)?
2. Your organization mandates the use of the sensitivity silos. Does it approve the use of one centralized identity management solution bridging all of the sensitivity silos?
3. If you cannot bridge the sensitivity silos with one solution, the only option is to treat each silo as a separate organization. Will your organization change its policy on the single centralized identity management system to allow bridging in the future?
4. Does your organization have a high staff turnover, or have a large number of contractors or out sourced staff?

5. Is your organization large or does it have multiple geographies that are self managing?
6. Does your organization already have a centralized or metadirectory in place or is it planning one?
7. If your organization is already using the DAC model with resource owners/administrators managing the identities of users, you could use a centralized solution to imitate this or you could move to an RBAC solution. Do you want to see further ROI and increased security?
8. If you chose to fully implement an RBAC model, will the political and business structures within your organization fully support the design work involved?
9. DAC Design selected?
10. RBAC Design selected?
11. Implement a single centralized identity management system with users assigned access rights based upon their approval to access one or many sensitivity silos. This is a simple form of RBAC with one role per sensitivity silo. It would be possible to make the silo model more granular, but this may detract from the essentially simple nature of the implementation. It should be noted that a user with access to one silo will gain access to all information within that silo; therefore, in its purest form, this architecture does not address the issues of Privacy or “Need to Know” management.
12. You can implement an identity management solution in each sensitivity silo, but should your organization’s policy change, you will be able to place a master Identity Manager over the existing silo Identity Managers to gain maximum ROI. You should therefore select a centralized management solution that is capable of supporting a hierarchy of identity management systems.
13. Treat each sensitivity silo as a discrete problem and analyze the RBAC/DAC requirements for each silo.
14. This selection is DAC. Make sure that the centralized identity management tool you selected has the capability to securely delegate the administration of users to the resource owner through an interface that does not require onerous training nor does it need a thick client to be distributed. Administration of the users should be delegated to the owners of the resources. Delegated resource control should be in line with corporate policies. Centralized audit for non compliance reports should be submitted to the resources owner regularly for their action.
15. After deployed, assistance should be given to those business units that want to develop an RBAC model within their “Owned” space. In addition, maintaining up-to-date business cases and continuing to win greater political influence for the RBAC model should be attempted.

16. Has sufficient political ground been gained to implement an RBAC model?
17. Your organization chose to use DAC, which will not allow for some of the ROI traditionally associated with RBAC. Other product features also show savings, however, and you should favor products with good feature/function coverage in these areas.
18. Workflow processing. The automation of the business processes for new hires and so on should be seen as a priority. Reducing the waiting time for provisioning new users will reduce productivity losses.
19. Even though DAC is the organizational model, it may still be possible to make savings by using limited or default roles. For example, every new user would automatically get LAN and e-mail accounts set up, while other systems remain within the purview of the resource owners.
20. Has a period of more than 12 months passed since you last checked the identity management system design?
21. Have any major infrastructure changes within your organization's operational systems taken place?
22. Has the nature of the external threat you face as an organization changed significantly?
23. A change occurred within your operating environment or a long period of time passed since you last validated your identity management system decisions. Run through the algorithm again to check on your design and amend it, if appropriate.
24. You selected a very simple type of RBAC to map onto the MAC model in place within your organization. This means that you will also be placing increased reliance upon the nature of your personnel and the vetting processes applied to them. It is possible to improve the silo granularity, but it will take time to design this granularity. Other software and hardware involved with privacy management and networking, for example, may already be in use within your organization. These should be factored into any design and planning for the solution.
25. The selection flowchart seems to suggest that you will be treating each of the sensitivity silos as a discrete identity management problem, but that you may in the future get approval to bridge the silos. The suggested method is to use a hierarchy, but if budgets and operational requirements allow, you could also scrap the existing system and replace it with a single central identity management model.
26. Reaching this point in the flowchart meant that owing to political limitations within your organization, you were forced to use the DAC model rather then the RBAC model, which you might naturally have selected. Using DAC, however, should be seen as a stepping stone towards RBAC. In simple terms, allowing the business owners to use the system may enable them to

create roles for their own systems. It may be possible to consolidate these local roles into larger ones as time passes.

27. As you move into the real design and planning work involved in an RBAC scenario, many of the “customer” business units are asked for their input into the role design problem. It may only be at this point, that “customer” business units realize exactly the impact of what you are proposing upon their “rights” to manage their own systems in their own way, regardless of the organizations security policies or of the costs involved. If this happens, you should return to question 8 and answer that question.
28. The DAC was selected and the focus on methods (other then RBAC) of saving costs. You should not lose sight of the fact that having a central tool also brings centralized audit capabilities that will improve the security of an organization. This risk mitigation, while difficult to quantify, still improves the viability of a business.
29. Wait one month before continuing. This ensures a revitalization of your identity management strategy every month. The length of time chosen should be less than one year, but is at the discretion of your organization, taking into account all of the threat/risk/resource issues you face.

17.6.2 Roles versus groups

One of the difficulties identity management system designers are facing is the way in which the terms groups and roles are used, often interchangeably or without a true understanding of their significance. They are defined as follows:

Roles A role describes the relationship or function of an individual to an organization. Resource(s) relates to the role and not to the individual.

Groups A group is specific to a target resource. It contains a subset of the users provisioned to that resource and grants access rights to a part of the resource.

Many identity management systems allow the users to be assigned to roles and hence provisioned to services. In addition, they can also provision users directly to services complete with group membership.

Even if the managed platform (the service) is working with a “group” concept, each group can be defined as roles and then each one has the correct permissions assigned. This can lead to some confusion or to much roles, so start simple is a must.

For example, let us say you want to do some RBAC for a LDAP-enabled directory Server. First you will define a role at the identity management solution and then assign this role to people. Then you create a group at this

LDAP-enabled directory that represents also the role already defined in the identity management solution. Access Control for the role is defined at the LDAP-enabled directory. Provisioning of people to this group is done when you define a provisioning policy that has the role (provisioning policy membership) assigned to the LDAP-enabled directory service with the corresponding group in the group membership attribute (provisioning policy entitlements). The identity management solution will take care of assigning the people that has this role to the corresponding group (provisioning and modification cycles).

Figure 17-4 shows the relationship between users, roles, services, and groups.

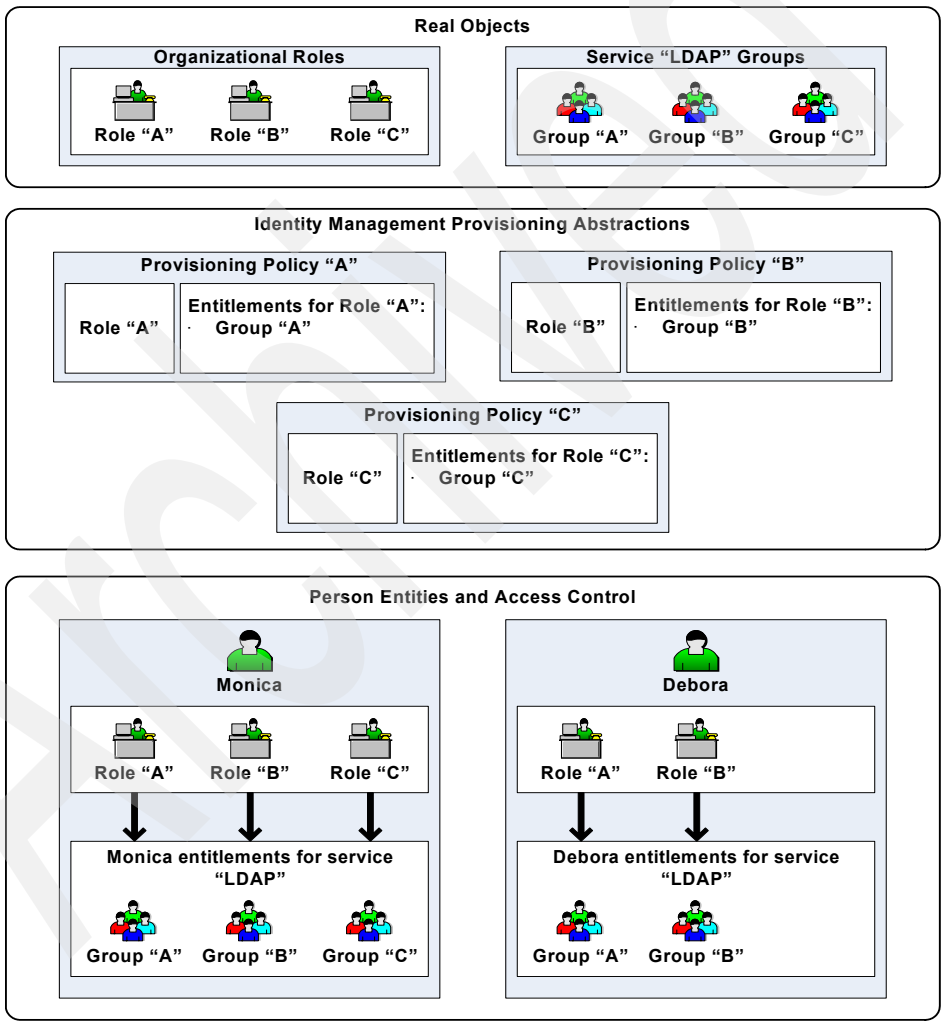


Figure 17-4 User/Role/Service/Group relationships

You can therefore use these systems to merely provision users directly to services, which is done in the absence of a valid RBAC design or in the case of the use of pure DAC.

You can also design the RBAC system such that one service is represented by one role. If each role represents only a single application, OS, database, and so on, then it is technically still an RBAC system, but it is functionally closer to a DAC system. This model is sometimes found within organizations that have not been able to successfully overcome the underlying politics. They can therefore claim to have upset no one and to have implemented a full RBAC system. The down side to this is that you spent the time and resources on implementing an RBAC system that will not deliver the expected ROI. This model is therefore pointless and not recommended unless political considerations are more important than cost concerns.

17.6.3 Designs

The process of designing an RBAC system is fairly straight forward.

If we had only two entitlements to access ("entitlement 1" has service "Corporate Directory" with group A and "entitlement 2" has service "Corporate Directory" with group B), then users could be placed into one of three roles: Role 1 (entitlement 1 only), Role 2 (entitlement 2 only), and Role 3 (entitlement 1 and 2).

In summary:

To access two entitlements, the number of possible roles are three:

- One role containing two entitlements
- Two roles containing one entitlement

Similarly, to access three entitlements, the number of possible roles are seven:

- One role containing three entitlements
- Three roles containing two entitlements
- Three roles containing one entitlement

To access four entitlements, the number of possible roles are 15:

- One role containing four entitlements
- Four roles containing three entitlements
- Six roles containing two entitlements
- Four roles containing one entitlement

As the number of entitlements increases, so does the potential number of roles. By the time twenty entitlements are required, there are 1,048,575 possible roles. It is clearly not practical to create all the possible roles and populate them. We

must reduce the number of roles to those required rather than to all those possible.

It seems that a common sense approach would be to list all the user repositories and then to list all the users along with their account requirements. An example of this kind of approach is shown in Table 17-2.

Table 17-2 User to repository mapping

User	**Group membership at "Corporate Directory"**				
	Group "A"	**Group "B"**	**Group "C"**	**Group "D"**	**Group "E"**
Alwena	Yes	No	Yes	Yes	No
Brian	Yes	No	No	Yes	Yes
Claudette	No	Yes	No	No	No
Daphne	No	Yes	No	No	No
Elizabeth	Yes	No	No	Yes	No
Francesca	Yes	No	No	Yes	No
Geoff	No	Yes	No	No	No
Helen	Yes	No	No	Yes	No
Ian	Yes	No	No	Yes	No
Jolina	Yes	No	Yes	Yes	No
Katya	Yes	No	No	Yes	Yes
Lupe	No	Yes	No	No	No
Mike	Yes	Yes	Yes	Yes	Yes
Neil	Yes	No	Yes	Yes	No
Ondine	No	Yes	No	No	No
Peter	No	Yes	No	No	No
Queenie	Yes	Yes	Yes	Yes	Yes
Ray	Yes	No	Yes	Yes	No
Sarah	No	Yes	No	No	No
Thomas	Yes	No	No	Yes	Yes
Uist	Yes	No	No	Yes	No

User	Group membership at “Corporate Directory”				
	Group “A”	Group “B”	Group “C”	Group “D”	Group “E”
Vera	No	Yes	No	No	No
William	Yes	No	No	Yes	Yes
Xerxces	Yes	Yes	No	Yes	No
Yvette	Yes	No	No	Yes	No
Zach	Yes	No	No	Yes	Yes

Grouping these roles into similar access requirements reveals that there would be six logical roles. So in this example, five entitlements give rise to six roles instead of all 31 possible roles, as shown in Table 17-3.

Table 17-3 User to repository mapping with roles

User	Role	Group membership at “Corporate Directory”				
		Group “A”	Group “B”	Group “C”	Group “D”	Group “E”
Elizabeth	Basic	Yes	No	No	Yes	No
Francesca	Basic	Yes	No	No	Yes	No
Helen	Basic	Yes	No	No	Yes	No
Ian	Basic	Yes	No	No	Yes	No
Uist	Basic	Yes	No	No	Yes	No
Yvette	Basic	Yes	No	No	Yes	No
Mike	CEO	Yes	Yes	Yes	Yes	Yes
Queenie	CEO	Yes	Yes	Yes	Yes	Yes
Claudette	Customer	No	Yes	No	No	No
Daphne	Customer	No	Yes	No	No	No
Geoff	Customer	No	Yes	No	No	No
Lupe	Customer	No	Yes	No	No	No
Ondine	Customer	No	Yes	No	No	No
Peter	Customer	No	Yes	No	No	No
Sarah	Customer	No	Yes	No	No	No

User	Role	Group membership at "Corporate Directory"				
		Group "A"	Group "B"	Group "C"	Group "D"	Group "E"
Vera	Customer	No	Yes	No	No	No
Xerxces	EMP & CUST	Yes	Yes	No	Yes	No
Alwena	HR	Yes	No	Yes	Yes	No
Jolina	HR	Yes	No	Yes	Yes	No
Neil	HR	Yes	No	Yes	Yes	No
Ray	HR	Yes	No	Yes	Yes	No
Brian	System Admin	Yes	No	No	Yes	Yes
Katya	System Admin	Yes	No	No	Yes	Yes
Thomas	System Admin	Yes	No	No	Yes	Yes
William	System Admin	Yes	No	No	Yes	Yes
Zach	System Admin	Yes	No	No	Yes	Yes

This is fine for 26 users and five entitlement, but the next problem that emerges is one of scale. The mere collection task involved for 1000 users and a larger range of entitlements becomes costly and, in larger cases, unrealistic. What is needed is a single data source that is collected automatically and contains all user/entitlement information, which can be used for reporting and analysis. Many centralized identity management solutions provide this kind of collection and reporting facility. As we saw in an earlier section, one way of countering the political objections to RBAC is to implement centralized identity management and progress towards RBAC as political support is developed. Once again, deployment of a centralized identity management solution can be used as a tool to develop a design for an RBAC model prior to the deployment of the RBAC model itself.

Following are a few other things to be careful of:

- No matter how you collect the information, it has to be correct at the point of collection. Examination of the user information in Table 17-3 on page 535 suggests that Queenie and Mike both have identical roles, in this case, CEO.

In practice, however, Queenie has the full access because she is the CEO, while Mike has been with the organization since leaving school and acquired a number of access permissions, as he has moved jobs within the organization and his access rights have not been rescinded. He is not the CEO.

- Similarly, Uist and Yvette both have the basic role, but neither worked for the company for over a year. Both these cases highlight the need to carry out a reality check audit as part of the process of designing an identity management system (whether or not it is RBAC).
- Some entitlements may have no IT dependencies. If a service is provisioned and the provisioning results in the involvement of a physical process (smart card generation and issue, uniform manufacture, and so on), then care must be taken not to include these potentially time delayed tasks into a workflow, which could delay other provisioning requirements. An RBAC design should take this type of entitlement into account.
- Up to now, we talked about each entitlement as a group of a service as though it has no relationship with other services. If group's definition are up to each platform administration, you have a higher potential for a high number of roles. If you can, however, do group definitions as "corporate groups" you will lower this potential and basically each platform administration will have to create the already-defined groups and then assign them correct access.
- Xerxces seems to be in a role of one person. He has picked up this unique role because he is both a basic employee of the organization and he is also a customer. We must therefore check with the security policy to see if he is allowed this "double" role under one name. It makes sense in some organizations to specifically separate Basic and Customer roles and disallow the Emp & Cust role.
- Even if an immediate RBAC design cannot be achieved, some roles should be self-evident. A basic corporate employee user (with network and e-mail access) and an eCustomer role (with e-business application access) are examples. Implementation of these roles will stimulate the RBAC design process and reduce the scale of the problem.

In practice, given the likely scale of most RBAC designs, it is necessary to include costing associated with the collection, clean up, and analysis of the existing user/repository data. It is a strong recommendation that any centralized identity management solution chosen should be capable of being deployed as a tool to help with the design of the full RBAC model. While this RBAC design is in preparation, some ROI can be gained from the automation of user provisioning and workflow processes.

17.6.4 Observations

Most enterprises use a blend of access control models based on the sensitivity of the information or the level of effort required to change the applications. Ideally the enterprise should have a predominant access control model such as RBAC and use the other access control model to handle exceptions. As a rule of thumb, the 80/20 ratio may be used. However, this ratio will vary based on the enterprise's business policies and security policies.

17.7 Planning the approach to the solution

In this section, we discuss the approach for architecting an identity and credential management system as being part of an overall enterprise security architecture, as well as the aspects of understanding and re-engineering enterprise business processes for managing identities and credentials.

Beyond the capabilities that a provisioning solution provides to deliver return on investment, it must also succeed in the complex, operational environment of an organization. The provisioning solution interfaces with a number of external systems and operates on a considerable amount of information distributed widely across the organization. It is important that the features of the provisioning solution be built on architectures and deployed in an environment appropriate to the organization.

Background information about how to architect an enterprise security architecture can be found in Chapter 2, "Common security architecture and network models" on page 19, which is based upon use of the IBM Method for Architecting Secure Solutions (MASS) covered in depth in Appendix A, "Method for Architecting Secure Solutions" on page 947. More about business process management can be obtained from the IBM Redbooks *Continuous Business Process Management with HOLOSOFX BPM Suite and IBM MQSeries Workflow*, SG24-6590.

17.8 Implementation plan

Any implementation should be part of a project and follow a standard set of steps or phases. There may be a number of methodologies involved, such as a project management methodology and one or more design and implementation methodologies.

We are concerned with the architecture and design for an identity management solution. The project to produce the architecture will normally follow one or more of the following:

- A company conducts an enterprise-wide Software Architecture project to review the entire IT environment and produce an enterprise architecture. The resulting architecture may dictate the need for a solution around identity management.
- A company conducts an enterprise-wide Security Architecture project. The project will look at all security aspects of the enterprise (not just the IT security). The resulting architecture will identify the security areas where the enterprise needs to focus. This may include identifying the need for a solution around identity management. This exercise should contribute to the enterprise Security Policy document that dictates the security policy to be applied to an enterprise, its employees, and its customers.
- A company purchases an identity management solution based on specific business needs, such as cost-cutting, audit compliance, and consistent application of corporate policies.

These leads to a project to deploy an identity management solution, which includes developing a product-centric architecture and design document. This is shown in Figure 17-5 on page 540.

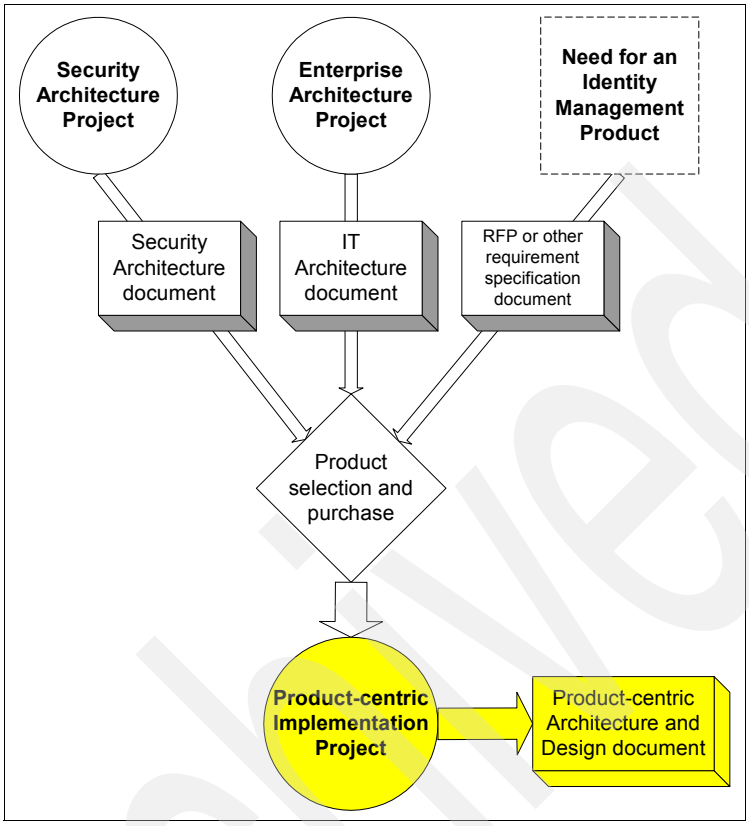


Figure 17-5 High-level and product-specific projects

Most projects involve business tasks (such as cost-benefit analysis and budgeting), project management tasks (such as scheduling, resource allocation, and risk management), and technical tasks (such as design and build). We restrict our discussion to the technical tasks associated with the production of the architecture and design document. Figure 17-6 on page 541 shows a set of generic steps or phases that relate to the architecture and design document.

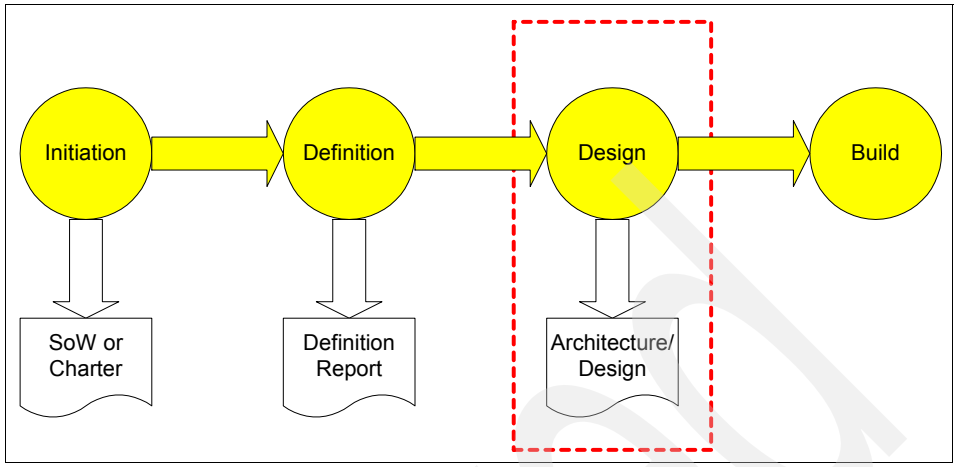


Figure 17-6 Generic implementation phases for a project

The steps are:

1. **Initiation**: This step normally involves identifying the project background and requirements at a high level. The deliverable for this step is some sort of Statement of Work (SoW) or Project Charter. The high-level requirements come from a preceding project (such as an IT architecture or security architecture project) or the software purchase requirements.
2. **Definition**: In this step, the project is defined in detail. This involves gathering data about existing systems, users, procedures, and other information and the detailed requirements of the solution. The deliverable for this step is one or more documents defining the project. These may include a Project Definition Report, a Requirements document, a Functional Specification, and an Existing System Analysis document.
3. **Design**: This step involves designing the solution. The deliverable for this phase is the Architecture and Design document.
4. **Build**: This is where the solution is built and implemented.

17.8.1 Definition of an identity management solution

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

The initial project definition is based on the documentation that triggered this project, such as the IT Architecture, Security Architecture, RFP, or equivalent.

These documents identify the business background, the business need for the solution, and, normally, the business and technical requirements for the solution.

For an identity management solution, the following areas must be defined in this phase (in no particular order):

- User management procedures: The procedures for managing users, who manages users, and what is required of the solution for managing users
- Password management procedures: The procedures for managing account passwords, who manages passwords, and what is required of the solution for managing passwords
- Access control management procedures: The procedures for managing access control, who manages access control definition, and what is required of the solution for managing access control
- Security policy: What the corporate security policy defines for users, accounts, passwords, and access control
- Target systems: The current system environment (including operating systems, databases, applications, the network, firewalls, physical location, and access control) and the system requirements of the solution
- Interfaces: The interfaces to the current identity management mechanisms and procedures and the integration requirements of the solution
- Auditing and reporting procedures: The procedures for auditing and reporting, who is involved in the auditing and reporting of users and their access, the audit requirements for the solution, and the reporting requirements for the solution
- Technical requirements: The other technical requirements for the solution, such as availability and recovery

Gathering this information normally involves a series of interviews and workshops with the people and teams involved in identity management. This may include the CIO, IT executive, security management/administration team, operations, help desk, key technical teams (UNIX sysadmin, Windows admin, and so on), and any application development teams and business managers involved in the project. The combination of these interviews and workshops develop a picture of how the system currently works and how it could be improved. The project owners should drive the requirements for the proposed system, although others may contribute to an understanding of the need for the requirements.

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

17.9 Business processes and identity management

The identity management solution comprises both business (or procedural) and technical (security subsystem-specific) functionality. An implementation involves installing an identity management tool, which could include integration with existing business procedures and perhaps some business process re-engineering (BPR). Both technical (product-related) and business (process-related) skills are required in the definition and design phases.

To produce an effective identity management solution, the architect must understand all identity processes involved in detail. Let us look at an example.

A new employee starts working for a company. How is the employee's identity information get created? Is there an HR database involved? How is that connected to salary and benefits? How does HR tie in with the IT department? How does that person get access to the applications needed to do the job?

The list of processes can include:

- A person joining a company and being defined to the HR system
- A person getting accounts to access applications
- A person getting passwords to use the accounts
- A person changing departments with bulk account changes
- A person changing a role with subtle account changes
- A person changing a surname and affecting accounts
- A person changing passwords
- A person resigning and being "marched out," requiring locking of accounts
- A person resigning but others need to access their account
- A password being reset by an administrator
- A locked account being unlocked
- An account being locked
- All accounts for a user being deleted
- A set of accounts being moved from one system to another
- An access control group being changed and affecting a number of users

This list is just a sample of the business process review exercise that should be performed as part of the Project Definition phase.

Implementing an identity management solution may involve designing a solution that complements the existing business processes or it may involve significant

business process re-engineering. The project requirements will indicate the level of business process re-engineering.

Adoption of any re-engineered processes must involve analysis of the impact of the solution on:

- The system owners. For an identity management solution, this will be the company executive (for example, the owners of the security policy) and the IT Security department.
- The system administrators. For an identity management solution, this will be the security administrators, help desk staff, and technical support.
- The system users. For an identity management solution, this will be everyone defined as IT users in an organization.

Any changes to processes could potentially affect every person in a company. These changes may drive the implementation of an identity management system (for example, reducing password-reset help desk calls by allowing users to change their own passwords). If there are to be changes to the processes, the architect and project team need to be cognizant of:

- Usability: Users of various skill levels may be using the solution, so the usability of the components must be appropriate to all levels of users.
- Documentation: Process changes affecting a large number of users will require greater documentation support than a change affecting a small team. This may include procedure documents, intranet pages, and online help.
- Education: As with documentation, if you are deploying significant changes to a large number of people, thought must be given to the education plan.

17.10 Conclusions

This chapter has examined the issues and circumstances that affect the design of an identity management solution.

IBM Tivoli Identity Manager, IBM Tivoli Identity Manager Express, and IBM Tivoli Directory Integrator are principally deployed to consolidate, provision, and manage users and identities across disparate identity groups, domains, and federated application repositories, regardless of whether these exist as singular or multiple management realms. Frequently, these realms inherit or adopt a commonly used role-based sphere of authority, whether hierarchical or scope of authority. Some of the goals for centralized management are:

- Easing compliance with security audits
- Consolidating control of the user management processes

- Eliminating inconsistencies from human error and “management by mood”
- Reducing training costs and education requirements
- Reducing help desk and overall administration costs
- Involving fewer people in day-to-day management
- Dividing work along organizational or departmental structures
- Improving response to user changes
- Leveraging user information in all business processes

Tivoli Identity Manager and Identity Manager Express work to address this structured approach to user and ID management by allowing a very high degree of configuration to map functional roles and associated access control provisioning to an organization’s IT business processes.

Tivoli Directory Integrator enables you to integrate many different applications either in conjunction with Tivoli Identity Manager or on its own. Directory Integrator offers a rich and easy set of tools that can help you get users, systems, and applications online and productive quickly.

Combining Tivoli Identity Manager and Identity Manager Express with Directory Integrator provides a robust and complete identity management solution. This solution can provide lifecycle management (user self-care, enrollment, and provisioning), identity control (access and privacy control, single sign-on, and auditing), and identity foundation (directory and workflow) to effectively manage internal users, as well as an increasing number of customers and partners through the Internet.

18

Identity Manager structure and components

This chapter provides information about the structure and components of IBM Tivoli Identity Manager. We discuss the concept of lifecycle management and IBM Directory Integrator's place in the context of identity management. Other topics include:

- The high-level logical component architecture for IBM Tivoli Identity Manager
- The various internal modules and sub-processes of Identity Manager
- Role-Based-Access-Control

18.1 IBM Tivoli Identity Manager entities

Identity Manager's role is to manage users and their accounts. Passwords, group memberships, and other attributes are associated with the users and accounts. These all relate to managed systems and applications. To enable management of users, accounts, and associated information, Identity Manager uses an organizational tree and roles, ACLs, and policies. Identity Manager also contains workflow, audit logs, and reports. These are described in the following sections.

The entities managed by Identity Manager are:

- Users, accounts and attributes
- Passwords
- Group memberships
- Managed systems and applications

18.1.1 Users, accounts, and attributes

A user can be classified as a *Person*, a Business Partner Person (*BPPerson*), or *custom Person*. A Person is typically an employee of the company or organization. A BPPerson is typically an individual who needs access to an organization's managed system or application but who is not considered an employee. All classes of users are managed in the same way. However, more information is required when adding a Person than when adding a BPPerson when using Identity Manager standard entities. A custom Person is used when the standard Person definition does not suit an organization and has to be extended for the organization.

Figure 18-1 on page 549 shows the relationship between a user, person and account. In the figure, a person, or an employee, Jane Doe is defined in the HR system of the company. When Jane is defined to Identity Manager, user accounts on managed resources such as UNIX, Microsoft Active Directory, and IBM Tivoli Access Manager can be provisioned according to the provisioning policy.

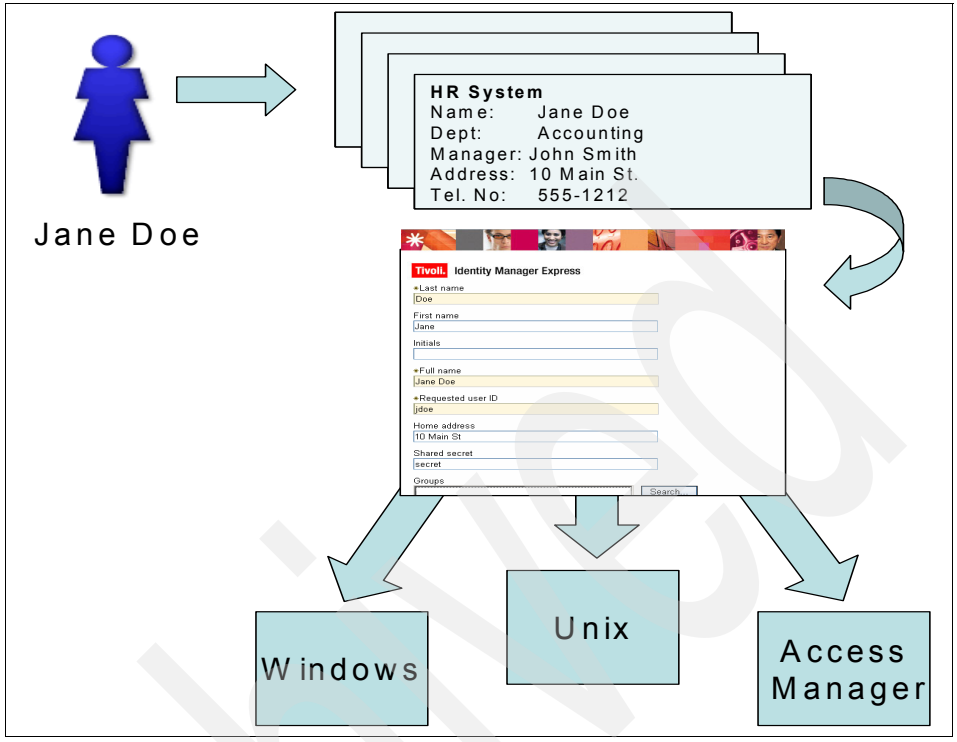


Figure 18-1 Person, user, and account relationship

A person can be located anywhere in the *organization tree*, so the organization tree represents the user structure of a company.

The personal information is defined as attributes on the person objects. This may include first, last, and full names, phone numbers, employee number, supervisor, and e-mail address.

An account is a person's access to Identity Manager or to a Service (managed resource), such as Linux, Active Directory, Solaris, SAP, and so on. Accounts have attributes that are defined by the managed resource.

Reconciliation is the process of determining the accounts existing at particular managed resources and processing each against provisioning policies defined within Identity Manager based on the owner.

An orphan account is an account that is not associated with a Person. Orphan accounts are generated when the reconciliation process cannot automatically associate the account with a person.

18.1.2 Identity feed

Identity Manager users are created either by importing identity records with the use of an *identity feed* or by manually creating each user. An identity feed is the process of synchronizing the data between an authoritative data source, such as an HR system, and Identity Manager. The initial reconciliation populates Identity Manager with new users, including their profile data. A subsequent reconciliation creates new users and also updates the user profile of any duplicate users that are found.

18.1.3 Passwords

All accounts have passwords. Account passwords can be centrally managed by their owners or administrators using the Identity Manager Web interface.

Password management is a very important topic, since passwords represent access to corporate applications, they have to be securely managed in their whole lifecycle. IBM Tivoli Identity Manager provides a full set of features to manage the passwords in a secure environment.

There are two options for application's passwords managed by IBM Tivoli Identity Manager, passwords can be synchronized or not. The synchronization can be applied to *all accounts* associated with a user or with *selected accounts*. For most passwords, this is a one-way synchronization. Identity Manager sets the password and pushes it to the managed targets. Identity Manager cannot accept a password change request from a target and push this to all associated accounts. The exception to this is when there is a password synchronization function for the platform, which intercepts a password change on the managed platform and passes it through Identity Manager.

When the password synchronization property is enabled, there is only one global password for all the applications managed by Tivoli Identity Manager. If an account is being set up for first time, password synchronization does not apply; there is only one account, and therefore, one password.

If a user has more than one account, password synchronization affects the following user or administrator actions:

- Creating a new account
- Changing a password for an existing account
- Provisioning an account
- Resetting an expired or forgotten password for an existing account
- Restoring an account that was suspended

If you have enabled the password synchronization property, there is no way for an user to change the password of only one account. All accounts receive the password change. Without the password synchronization option enabled, users could select which account or accounts are to be changed. Administrators can always change passwords for selected accounts by using the service account management, but, this would imply that a user will have different passwords across platforms or applications, because the reconciliation process does not synchronize passwords.

There is a process where Identity Manager generates a random password. This can be displayed to an administrator or mailed to a user. Also, there is the option where Tivoli Identity Manager could generate a password for an account, and send a URL to the user for password pickup.

Tivoli Identity Manager uses a challenge/response function to verify a user's identity if they have forgotten their Identity Manager password. The challenge questions can be picked from a standard list or defined by the user. When a user logs into Identity Manager for the first time, they enter or select the challenge questions (if configured) and responses. On subsequent logins to Identity Manager, they can select a *forgot password* option and a subset of the challenge responses are used to verify the user.

18.1.4 Group membership

Accounts are given access on target systems and applications via some form of group membership. These may be groups on UNIX systems or Windows domains, SAP groups or profiles, or another access control grouping mechanism. Membership is granted by using a group attribute on accounts.

Group lists, for most managed targets, are updated with the reconciliation function. Thus, administrators do not manually enter group names; they select from a list that is in sync with the respective target.

Note that Identity Manager does not create or delete groups on managed targets. Nor does it manage ACLs or resource access on the managed targets. This must be performed by the local administrators or application owners using the native system or application tools.

18.1.5 Managed systems and applications

Identity Manager manages users on many managed systems. These include operating systems, such as many flavors of UNIX and Windows servers, and applications, such as databases and business applications.

Identity Manager deploys an adapter to perform the administration of accounts on the system or application. Some adapters are deployed to the system or application and interact locally. Others can operate remotely and be deployed anywhere in the network.

On the server side, there is a set of definitions of each type of resource to be managed; this set of definitions is called a *profile* or *service profile*. Each profile must be deployed to the Tivoli Identity Manager Server before you can manage this type of resource.

The adapter to server communication by default does not use SSL, however you can enable SSL communication with one-way or two-way authentication.

On the Tivoli Identity Manager Server, WebSphere Application Server SSL support is used.

Each adapter instance is defined as a *service* within the Identity Manager server. Accounts are associated with specific services. For example, there is a service for every Linux server. The services are defined within the organization tree and can have ACLs attached to control administrative access to functions performed against the service. A service can only be defined for a pre-existing managed target.

There is a *service profile* for every type of service. For example, there is one service profile for Linux services. The service profile defines the account attributes for that type of service.

18.2 IBM Tivoli Identity Manager management entities

Identity Manager uses the following entities for management:

- Organizational tree and roles
- Identity Manager roles and ACLs
- Policy
- Workflow
- Audit logs
- Reports
- Lifecycle management

These are discussed in the following sections.

18.2.1 Organizational tree and roles

Central to Identity Manager is the organization tree (or org tree). It defines the structure for the organization that Identity Manager is being deployed into. The tree consists of:

- An organization: There is normally only one organization at the top of the org tree.
- One or more locations: These are locations defined by the business.
- One or more organizational units: These are teams or departments as defined by the business.
- One or more business partner organizations: These are business partners as defined by the business.
- One or more admin domains: These are Identity Manager groupings for administration.

There is no technical difference between locations, organizational units, or business partner organizations. They use different icons and allow the org tree to be modelled as the administrators want.

All people are attached to the org tree at a single point.

A policy is attached to points in the org tree and can apply to objects at that level or to all objects at or below that level. This policy can control the provisioning of accounts, account user ID generation and password strength. Thus, you could have a corporate-wide password policy defined at the organization level in the org tree and a specific password policy that applies to a specific branch or department of the organization.

Identity Manager *roles* or *organizational roles* and ACLs are also attached to points in the org tree, defining the scope of specific access rights within the Identity Manager product.

Organizational roles are used to model job roles within an organization. They can be used to map users to a set of accounts that are granted through a *provisioning policy*. organizational roles can be *static* or *dynamic*. In static organizational roles, assigning a person a static role is a manual process, and it can be done every time it is needed, during an identity creation or through an identity feed.

Dynamic organizational roles set person membership to a specific role based on valid LDAP filters. Dynamic organizational roles are evaluated at different times:

- When a new person is created in the Tivoli Identity Manager system.
- When a person information change in the Tivoli Identity Manager system.
- When a new dynamic organizational role is created.

Every time a dynamic organizational role is evaluated, all people who fit the LDAP filter affected with the membership of the role, and their personal information is updated with the membership information.

18.2.2 Identity Manager groups and ACIs

A user's access within Identity Manager (for example, the functions they can perform in Identity Manager) is governed by the groups they belong to.

Identity Manager governs user access rights using Access Control Item (ACI). An ACI controls user access by defining the access privileges of an Identity Manager group or ACI principal. Members of an Identity Manager group or ACI principal can view and perform operations on attributes within a target class (context) as defined by the scope of the ACI.

This role-based access is for Identity Manager users assigned to the Identity Manager groups. Identity Manager system administrators are not controlled by ACIs because the administrator account, by default, has access to all functions in the system. All other users, by default, do not have access to any functions or features in the system.

18.2.3 Policy

Identity Manager employs four types of policy: provisioning policy, password policy, identity policy, and service selection policy.

Provisioning policy

A *provisioning policy* confers access to many types of *managed services* (Identity Manager, Windows 2003, Solaris, and so on) by granting a person access based on an organization (for example, a person's location in the org. tree, an organizational role, or all people not in any organizational role). In other words, access to a target managed service is either:

- Granted to all persons in an organization
- Granted only to persons assigned to a specified organizational role
- Granted to persons not covered by any other provisioning policies on any of the entitlement targets associated with the current policy

A provisioning policy is used to define what accounts can be created for a user and it can, optionally and automatically, create the accounts on those systems. It can also be used to define a specific approval workflow process that has to be applied to the accounts.

At the time of creating a provisioning policy, there are many accounts that could be affected by this new policy. In a test environment, it might not have major implications, but, in a production environment, it could have implications based on access granted because of the policy. To prevent any accidental behavior of a provisioning policy there is a *simulation function* that helps you understand what and who is affected by the creation or modification of a provisioning policy.

Service selection policy

A *service selection policy* extends the ability of provisioning policies by provisioning accounts based on personal attributes. In order for a service selection policy to be enforced, a provisioning policy must target it. The service selection policy then identifies the service type to target and defines provisioning based on a JavaScript.

Identity policy

An *identity policy* defines how a user's ID is created. Identity Manager automatically generates user IDs from the identity policy. Identity policies can be set as a global policy or as a service specific policy. If the identity policy is not a global policy, the policy can be assigned on a per service basis (for example, it only applies to specific service types) or it can be assigned to a combination of service types or instances. For example, if all user IDs must be composed of the user's first initial and last name, a global identity policy must be created for the organization. If all user IDs for a specific service must contain a certain number, a service specific identity policy must be created for the service.

Password policy

A *password policy* sets parameters that all passwords must meet, such as length, type of characters allowed and disallowed, and so on. You can set up password policies to apply to any of the following:

- Only one service instance or more than one service instances
- All service instances of only one service type or multiple service types
- All services, regardless of service type

18.2.4 Workflow

A workflow is a set of steps or activities that define a business process. You can use the Tivoli Identity Manager workflows to customize account provisioning and

lifecycle management, such as adding, removing, and modifying people and accounts in IBM Tivoli Identity Manager. The workflow process is defined by a workflow design. When a user places a request for a new account, new access rights, or changes to an existing account, the request must be approved by signature authorities defined by a workflow design.

A workflow design can be added to an entitlement in a provisioning policy when the entitlement is defined or at a later time. This helps you customize how resources (accounts, services and so on) are provisioned.

Workflow designs are built using the Identity Manager GUI. The design created by the visual programming Java applet in the GUI actually produces an XML implementation under the covers.

18.2.5 Logs and audit

Identity Manager employs logging features that log the events during specific transactions. This facilitates isolating and debugging of problems, focused on troubleshooting key Tivoli Identity Manager business processes, such as:

- Add, Modify, Suspend, Restore, Delete Person
- Add, Modify, Suspend, Restore, Delete Account
- Change Password
- Add, Modify, Delete Provisioning Policy
- Add, Modify, Delete Dynamic Role
- Add, Modify, Delete Service Selection Policy
- Reconciliation and Event Processing (including identities)

Reports can also be run against the audit logs. Any action taken by a Tivoli Identity Manager user that changes a business object or the configuration of the system is audited:

- ACI management
- User Management (People, Role and Container Management)
- Policy Management (Provisioning, Service Selection, Identity)
- Service Management
- Account Management
- Configuration Management
- Authentication Events

Only non-workflow actions will be audited (workflow actions currently have an audit trail). Workflow actions are also audited, but only at the high level. The audit log for any activity can be viewed using the Identity Manager Web user interface.

18.2.6 Reports

Tivoli Identity Manager provides several different standard reports. These reports use predefined templates that enable you to specify criteria that produce the report details that you want:

- Account Operations: Shows account activities.
- Account Operations Performed by an Individual: Shows account operations that have been requested by one or more individuals.
- Approvals and Rejections: Shows requests for which an approval activity has occurred.
- Pending Approvals: Shows the status of pending approvals for access to services.
- Operation Report: Shows requests by type of operation, date, who requested the operation, and for whom the operation was requested.
- Individuals' Accounts: Shows the account information of individuals.
- Individuals' Accounts by Role: Shows the names of individuals associated with a specified role.
- Entitlements Granted to an Individual: Shows entitlements that have been provisioned to individuals and the provisioning policies that govern the individuals.
- Policies Governing a Role: Shows roles and the policies that govern the roles.
- Suspended Accounts: Shows suspended accounts and associated persons and services.
- Suspended Individuals: Shows person names that have been suspended.
- Policy Report: Shows policies and related information, including associated services, organizational units, and roles.
- Dormant Accounts: Shows accounts with no activity.
- Access Control Items (ACIs): Shows ACI definitions and associated information.
- Summary of Accounts on Services: Shows service types and corresponding service names and accounts.
- Services: Shows requests for existing service instances.

- Reconciliation Statistics: Shows the status of various accounts and account activities.
- Rejected Report: Shows denied requests by date, who requested the operation, and for whom the operation was requested.
- User Report: Shows all operations that meet the specified criteria.
- Account Report: Shows people and their associated accounts and whether or not the accounts are in compliance with current policies.
- Audit Events: Shows audit records of user actions.
- Non-Compliant Accounts: Shows non-compliant accounts and associated services.

Access to any of the reports is defined by the report ACIs. These ACIs govern the availability of reports for all users, including access permission to view reports and access permission to run reports.

18.3 Logical component architecture

The logical component design of Identity Manager may be separated into the following layers of responsibility, which are shown in the center of Figure 18-2 on page 559.

- The Web User Interface layer
- The Application layer
- The Service layer

Figure 18-2 on page 559 illustrates the graphical workings of the package.

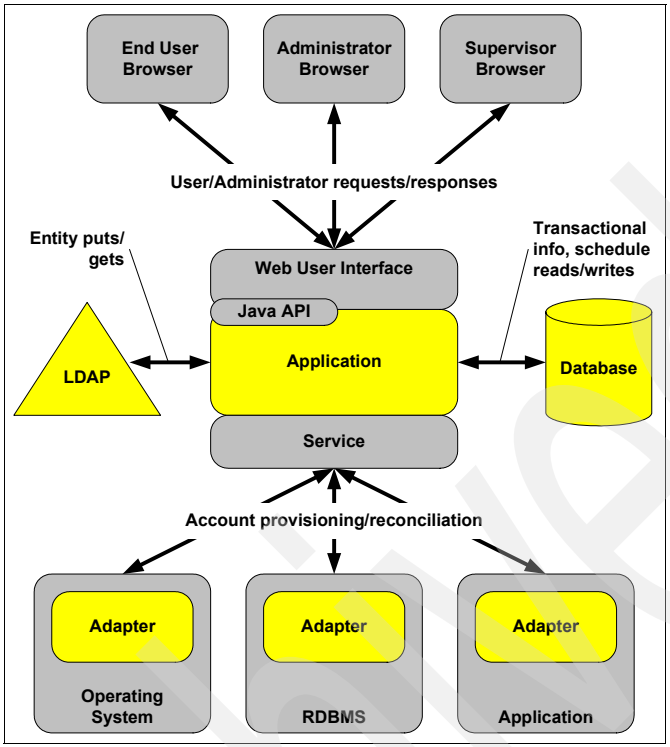


Figure 18-2 Identity Manager logical architecture

18.3.1 Web User Interface layer

The Web User Interface module is a set of combined sub-processes that provide the content to a user's browser and initiating applets (both run on the client and the server), such as the Workflow Design and the Form Creation. The Web User Interface is the interconnecting layer between that of the user's browser and the identity management Application layer.

In Figure 18-2, there are three types of user interaction points: user, supervisor, and administrator. These types are merely conceptual. Tivoli Identity Manager enables customers to define as many different types of users with different permissions as they like.

However, for this diagram, it is important to note that the system is built with a general concept of the capabilities of the system users. For example, it is assumed that the administrator needs advanced capabilities and requires a more advanced user interface, possibly requiring a thicker client (applet). It is assumed that the supervisor needs fewer capabilities but may still require concepts such

as an organizational chart. Because the number of supervisors in an enterprise will vary, a thick client is not practical. Last, there are no assumptions made for the end user. The interface presented to the end user must be a thin client with very basic and intuitive capabilities.

The Web User Interface subsystem contains all modules necessary to provide a Web-based front end to the applications of the Applications subsystem.

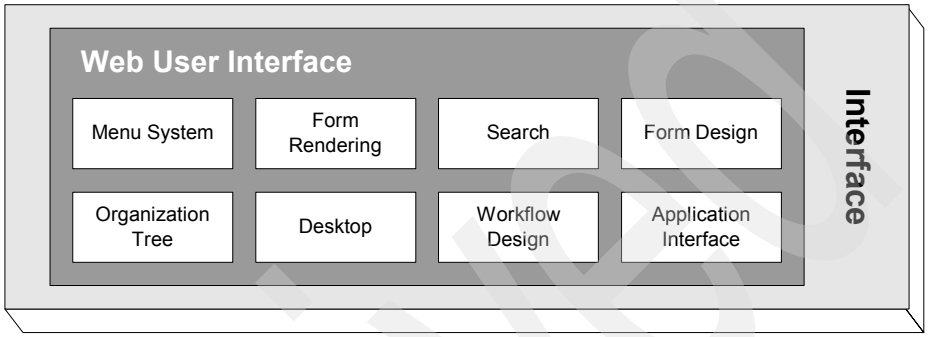


Figure 18-3 Web User Interface module subprocesses

18.3.2 Application layer

The core of the IBM Tivoli Identity Manager system is the Application Layer. Residing on an application server, the application layer provides the management functionality of all other process objects.

The Application subsystem contains all modules that provide provisioning specific capabilities, such as identity management, account management, and policy management. Each application makes use of the core services in the Services layer to achieve its goals. It is the Applications module that provides the external interface to the provisioning platform. Following is a brief description of each module, as well as a graphical overview, as shown in Figure 18-4 on page 561.

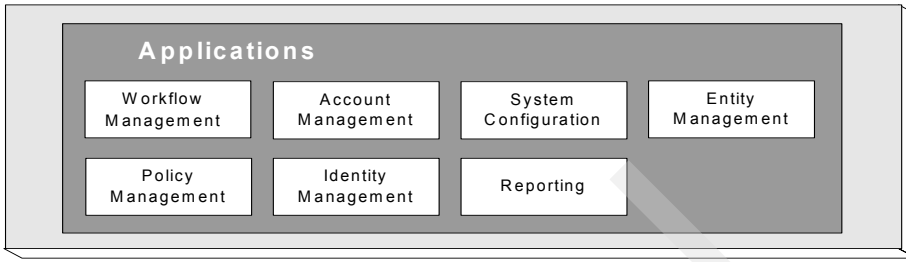


Figure 18-4 Applications module sub processes

Workflow Management module

The Workflow Management module provides the capabilities required to manage workflow processes, such as their addition, modification, and removal. The ability to view the status and details of active and historical processes is also provided in this module.

Policy Management module

The Policy Management module provides the capabilities to manage the policies in the system, including provisioning, \password, service selection, and identity policies.

Account Management module

The Account Management module provides the capabilities required to manage accounts, such as their addition, removal, suspension, reinstatement, and modification.

Identity Management module

The Identity Management module provides the capabilities required to manage identities, such as their addition, removal, suspension, reinstatement, transferal, and modification, including the changing of roles. The definition of roles, including dynamic roles, is also included in this module.

System Configuration module

The System Configuration module provides the capabilities required to manage the IBM Tivoli Identity Manager system itself, such as defining behavioral properties.

Reporting module

The Reporting module provides the canned report capabilities of the system. This module provides the query and formatting of the reports driven from the user interface.

Entity Management module

The Entity Management module provides the capabilities required to manage the types of entities managed by the system, such as types of identities and accounts. This includes the ability to define the schema for the entity type, the operations the entity type can support, and the lifecycle of the entity type.

Note: Sitting between the Web user interface and the Application layer in Figure 18-2 on page 559 is the public Java API. This API provides a set of Java classes that abstract the more commonly-used functions of the provisioning platform such as identity management, password management, and account management. The classes that make up this API are the same classes the Identity Manager product uses for its out-of-the-box user interface.

For more information, refer to documentation provided with the Applications API in the <ITIM_HOME>/extensions/doc/applications directory.

18.3.3 Service Layer

If the IBM Tivoli Identity Manager server is the application of complex rules that have been developed, then the applications server is the engine that runs those rules or objects. It is communicating not only to the user facing Web server, but also to the adapters residing on the managed services and to directories for storage of information.

The Core Services subsystem contains all modules that provide general services that can be used within the context of provisioning, such as authentication, authorization, workflow, and policy enforcement. These services often use other

services to achieve their goals. A brief description of each module is given in the following sections, as well as a graphical overview, as shown in Figure 18-5.

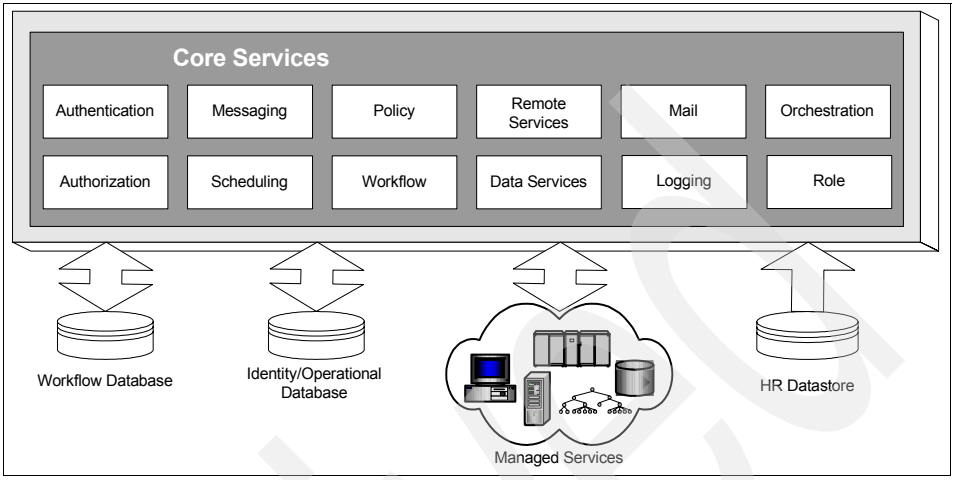


Figure 18-5 Core Services module sub-processes

Authentication module

The Authentication module provides a set of authentication implementations that can be used by clients of the service. Examples of these implementations are simple password authentication and X.509 certificate authentication. The module is designed as a framework that can be extended by customers to provide their own implementations.

Authorization module

The Authorization module provides an interface to enforce authorization rules as clients attempt operations in the system. These rules apply to accessing data within the system, as well as to operations that can be applied to the system data.

Mail module

The Mail module provides an interface for notifying users via a messaging system, such as e-mail. The module is configurable to accommodate different messaging systems.

Messaging module

The Messaging module provides guaranteed asynchronous messaging to and between internal modules in the architecture. The module relies heavily on the Java Message Service (JMS) specification to provide support for multiple messaging middleware vendor implementations.

Scheduling module

The Scheduling module provides a timer that notifies clients of timed events that they have subscribed for. The Scheduling module uses the Messaging module to notify those clients.

Note: The *clients* discussed in this section are internal to Identity Manager. For example workflow is a client to scheduling—it uses scheduling to allow workflows to start at a later date instead of immediately.

Policy module

The Policy module enforces the policies that associate users with services. The module ensures that provisioning requests conform to the policies that are defined. The module resolves the appropriate policies that apply to a user and determines the services for which that user is authorized. The module validates and generates passwords. The module generates identities for users and accounts.

Workflow module

The Workflow module executes and tracks transactions within the system. This would include the provisioning/de-provisioning of a service, a user's status change, the custom process associated with a provisioning request in the system, or any other transaction that affects a user's, or group of users', access to services. Each of these transactions is persistent for fault-tolerant execution and historical auditing purposes. Clients can query the Workflow module for the status of the transactions being executed.

Remote Services module

The Remote Services module provides the interaction with the external systems for provisioning and de-provisioning services. The synchronization of service information and user information is also performed within this module. The module is designed as a framework that can be extended by customers to provide their own implementations of provisioning and de-provisioning of services. This allows the platform to easily support different protocols and APIs that may be supported by the resources to be provisioned.

Data Services module

The Data Services module provides a logical view of the data in persistent storage (LDAPv3 directory) in a manner that is independent of the type of data source that holds the data. The model abstracts the details of the stored data into more usable constructs, such as Users, Groups, and Services. The model also provides an extendable interface to allow for customized attributes that

correspond to these constructs. Meta-Data information about the persistent data can also be retrieved using this module.

Logging module

The Logging module provides a common logging interface to all other modules. The implementation of this logging interface is provided by the JLOG logging toolkit.

Role module

The Role module evaluates dynamic memberships to roles. This module is called upon when an identity or dynamic role definition changes to identify which identities should be members of dynamic roles.

Orchestration module

The Orchestration module provides a coordination service for extensible operations that are performed on entities and manages the lifecycles of those entities. For instance, the orchestration module provides an abstraction layer to the Account Management application for executing the steps needed to provision an account of a given type. Regardless of the steps involved, which could be customized or changed, the Account Management module would always use the same interface to the Orchestration module.

18.3.4 LDAP directory

The IBM Tivoli Identity Manager system uses an LDAPv3 directory server as its primary repository for storing the current state of the enterprise it is managing. This state information includes the identities, accounts, roles, organization chart, policies, and workflow designs.

More details on the LDAP Directory and its schema are available in the *IBM Tivoli Identity Manager Database and Schema Reference Version 4.6,* SC32-1769.

18.3.5 Database

A relational database is used to store all transactional and schedule information. Typically, this information is temporary for the currently executing transactions, but there is also historical information that is stored indefinitely to provide an audit trail of all transactions that the system has executed.

More details on the database and its schema are available in *IBM Tivoli Identity Manager Database and Schema Reference Version 4.6,* SC32-1769.

18.3.6 Resource connectivity

The back-end resources that are being provisioned by IBM Tivoli Identity Manager are generally very diverse in their capabilities and interfaces. The IBM Tivoli Identity Manager system itself provides an extensible framework for adapting to these differences in order to communicate directly with the resource. For a more distributed computing alternative, a built-in capability to communicate with a remote adapter is provided. The adapters typically use an XML-based protocol, either Directory Access Markup Language (DAML) or Directory Service Markup Language (DSML version 2) or Remote Method Invocation (RMI), as a communications mechanism. Adapters are typically know as follows:

- ADK-based adapters

  An ADK-based adapter is a Tivoli Identity Manager adapter typically developed by Tivoli using the Tivoli Identity Manager Adapter Development Kit. These adapters usually use the DAML protocol.

- Tivoli Directory Integrator-based adapters

  A Tivoli Directory Integrator-based adapter is a Tivoli Identity Manager adapter developed and running on Tivoli Directory Integrator. These adapters usually use DSMLv2 or RMI protocols, being the RMI the new standard.

Directory Access Markup Language connectivity

DAML is a proprietary XML message format used when communicating with one of IBM Tivoli Identity Manager's standalone adapters. These adapters are programs installed on either the managed resource, or on a host that can manage the resource through a remote administration API.

DAML is a simple XML schema definition that enables the encoding of identity information in the form of an XML document so that it can be easily shared via IP protocols such as HTTP/S, as shown in Figure 18-7 on page 568.

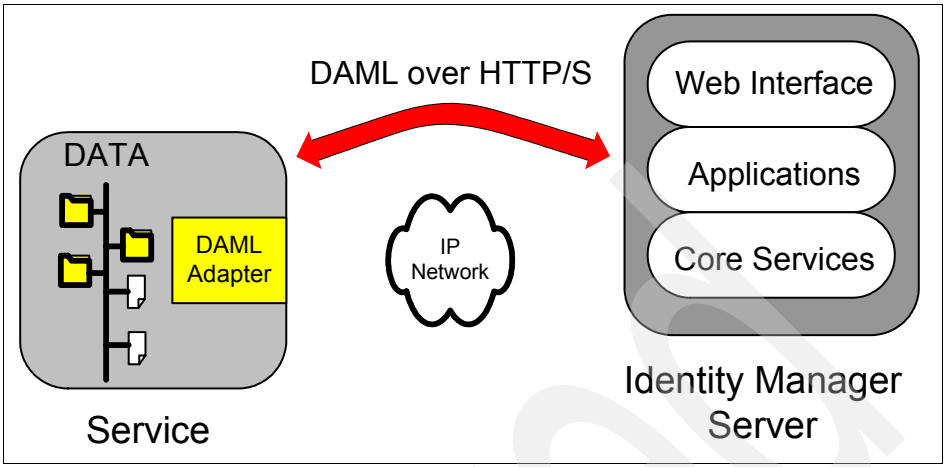


Figure 18-6 DAML connectivity to a service

Transactions from the IBM Tivoli Identity Manager server are sent securely via HTTPS to the service adapter and then processed by the adapter.

For example, if a service has just been connected to the IBM Tivoli Identity Manager server, the accounts that already exist on the server may be reconciled or pulled back in order to import the users' details into the IBM Tivoli Identity Manager LDAP directory. If a password change or a provisioning of a new user occurs, the information is transferred to and then processed by the adapter. The adapter deposits the new information within the application or operating system that is managed.

Directory Services Markup Language connectivity

DSMLv2 is an industry standard XML message format for the representation of directory data and operations. DSMLv2 is mostly used in conjunction with IBM Tivoli Directory Integrator to create custom adapters.

Directory Integrator provides an easy and flexible way to link IBM Tivoli Identity Manager to a wide variety of managed resources. Directory Integrator offers connectors that can be used to manage data in files, directories, databases, message queues, as well as many other data sources. It allows you to define, using simple scripts, how DSMLv2 operations issued by IBM Tivoli Identity Manager should be translated into operations on the managed resource.

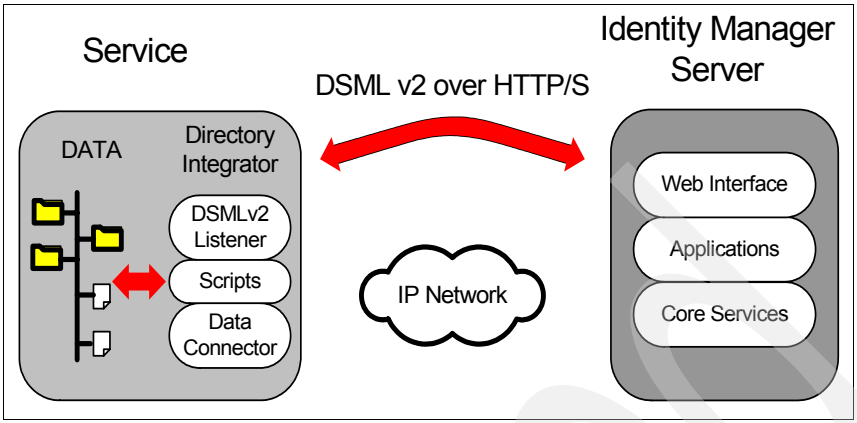


Figure 18-7 DSMLv2 service communication

Transactions from the Identity Manager server are sent securely via HTTPS to the service adapter and then processed by the adapter example of this would be if a service has just been connected to the IBM Tivoli Identity Manager server, the accounts that already exist on the server may be reconciled or pulled back in order to import the users' details into the IBM Tivoli Identity Manager LDAP directory. If a password change or a provisioning of a new user occurs, the information is transferred to and then processed by the adapter. The adapter deposits the new information within the application or operating system that is managed.

Other DSMLv2 usage is for identity datafeeds.IBM Tivoli Directory Integrator can be used to integrate identity sources such as HR, ERP, CRM, contractors systems, directories, databases and many other identity sources to feed person entities and keep them up-to-date at the identity management solution.

Remote Method Invocation connectivity

Additionally to DSMLv2, Tivoli Identity Manager can connect to Tivoli Directory Integrator adapters using Java Remote Method Invocation (RMI) calls.

Figure 18-8 on page 569 shows the communication between Identity Manager and a Director Integrator RMI-based adapter, possibly because of both products having Java RMI available as a communication service.

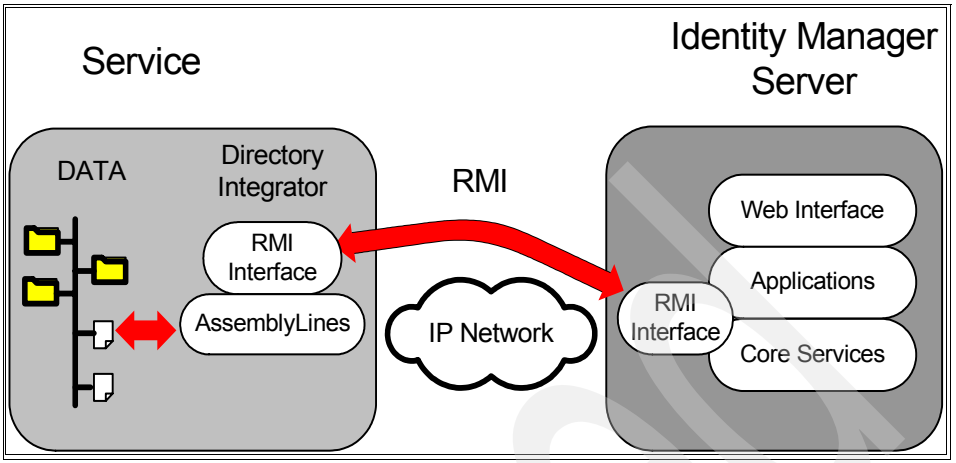


Figure 18-8 Tivoli Identity Manager and Tivoli Directory Integrator RMI communication

Identity Manager reverse password synchronization

Identity Manager allows for password changes to be initiated from some managed resources in addition to the standard password change screens within the Identity Manager Web application. For example, a Windows password change initiated via the native Windows mechanisms can be captured by the Identity Manager Windows Active Directory reverse password synchronization component which then sets the password directly to the Windows environment and have the request sent to the Identity Manager server for processing. This triggers a password change within Identity Manager and it is treated as a standard Identity Manager password change with the only difference being that it is not sent back to the originating managed resource. For example, in the case of Windows, Identity Manager will not send a request for the user's Windows password to be changed again.

The Identity Manager reverse password synchronization component can be configured to use the Identity Manager password policies to enforce password rules.

18.3.7 Lifecycle example

With so many pieces it can be difficult to understand how these work together. In the following example there is a view of major pieces and protocols.

In 17.5, "Lifecycle management" on page 523, we learned that Identity Manager performs lifecycle management for person entities and their accounts. Figure 18-9 on page 570 depicts a common example of how lifecycle management works.

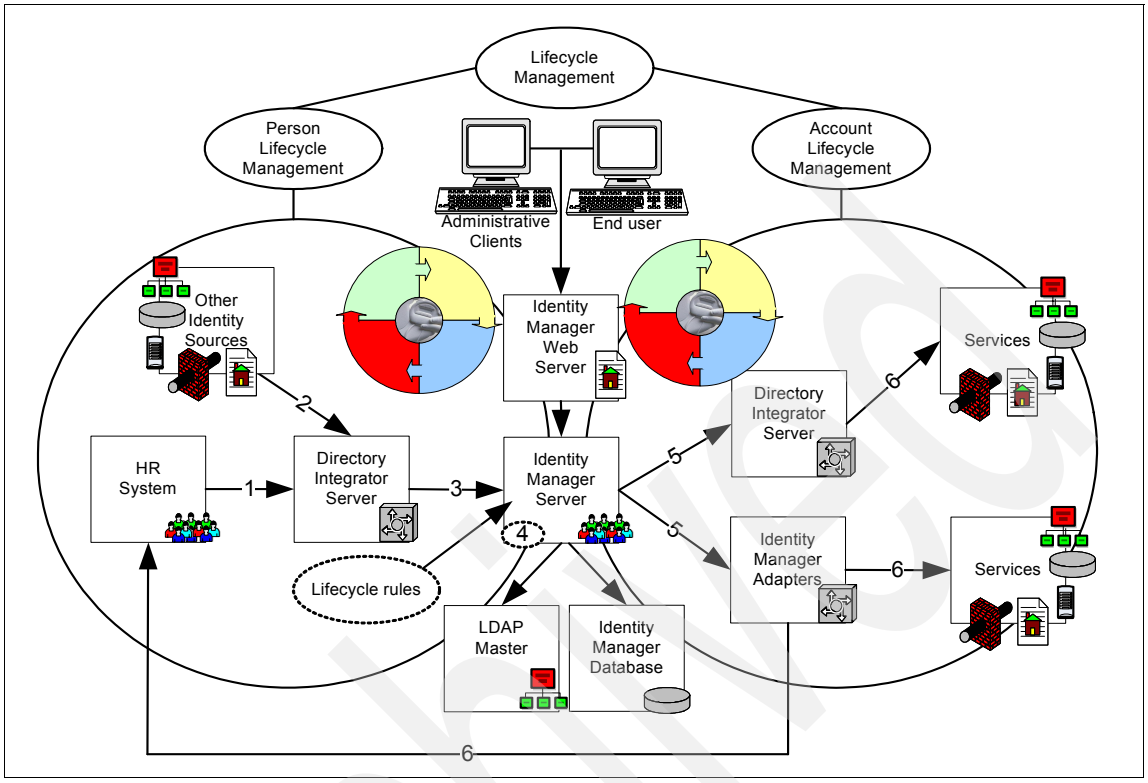


Figure 18-9 Lifecycle management scenario

Lifecycle management follows this flow:

1. A Directory Integrator AssemblyLine called *identity datafeed* retrieves identities from the HR system, also referred to as the *authoritative source for identities*. Depending on the company requirements and HR system technology, this can be a push or pull operation, based on schedule or even triggered from an HR identity modification.
2. Depending on the required attributes, the *identity datafeed* can pull additional data from other identity sources.
3. When identity data is ready, a DSMLv2 over http requisition with the person entity and it's operations is sent to the Identity Manager server.
4. The server looks for a current person entity in it's LDAP repository and creates a new entity or updates an existing entity if the person entity already exists. According to the changes, the Identity Manager server recomputes

user entitlements and starts each account provisioning or modification cycle, for example:

a. New person entities are created and accounts are provisioned.

b. Suspend operation suspends all accounts owned by this person.

c. Restore operation restores all accounts owned by this person.

d. Role changes triggers account provisioning, deprovisioning, and modifications.

e. Attribute changes triggers account dependent attribute modifications.

f. Delete operations delete person entity and suspends user accounts until some lifecycle rule deletes them.

Note that lifecycle rules may complement these operations but are not dependent on an identity datafeed. It means they can be triggered anytime for many different lifecycle management operations. The way operations are executed can also be customized according to business requirements, for example:

– Delete operations only suspend accounts.

– Delete operations also delete accounts.

5. Account lifecycle management operations are triggered based on person lifecycle management results. Provisioning, deprovisioning, and account modifications are triggered as separate subprocesses. In this example there are services managed by Tivoli Identity Manager adapters and Tivoli Directory Integrator custom adapters.

6. Each adapter knows how to manage its platform. The requests are executed, completing the current lifecycle management. Note that HR system accounts are also being managed by a Tivoli Identity Manager adapter. These accounts are for HR system management and are independent of HR data.

This scenario is discussed in more depth in 19.5, “Importing and synchronizing user data” on page 603.

18.4 Conclusion

As we have seen, lifecycle management is the process in which identities are completely managed and where the view of the process is most evident. To achieve a successful and complete automation process, the process itself must be broken down into the many components that make up the final result.

Identity Manager is a very powerful tool that has many layers and modules within its structure to enable a complete and successful solution for any environment.

By employing an extensible framework to adapt to the many available data sources and utilizing a standards-based approach, all of the resources in the back-end data stores become accessible and linked.

19

Identity Manager scenarios

This chapter provides real-world examples of an IBM Tivoli Identity Manager solution. Beginning with basic security architecture considerations and server placements in network zones. It strives to take you through the business drivers, concerns, and constraints you may encounter when implementing an identity management solution.

This chapter also shows you processes and considerations when integrating with other IBM Tivoli security packages as well as the use of IBM Tivoli Directory Integrator in a complex environment.

19.1 Basic security architecture considerations

There are several steps involved in transitioning component-level specifications into security subsystems. An IBM Tivoli Identity Manager system is part of an enterprise environment and because of that, the architecture has to be flexible enough to support different configuration options. This section discusses secure component placement that must be included when creating an identity management design.

19.1.1 Network considerations

Keep in mind that the network examples we are using do not necessarily include all possible situations. There are organizations that extensively segment functions into various networks. In general, the principles discussed here may be easily translated into appropriate architectures for such environments.

Placement of various Identity Manager components within network zones is a reflection of the security requirements of each organization. While requirement issues may often be complex, especially with regard to the specific behavior of certain applications, determination of a Identity Manager architecture that appropriately places key components is generally not difficult. With a bit of knowledge about the organization's network environment and its security policies, reasonable component placements are usually easily identifiable.

Figure 19-1 on page 575 summarizes the general Identity Manager component type relationships to the network zones discussed above.

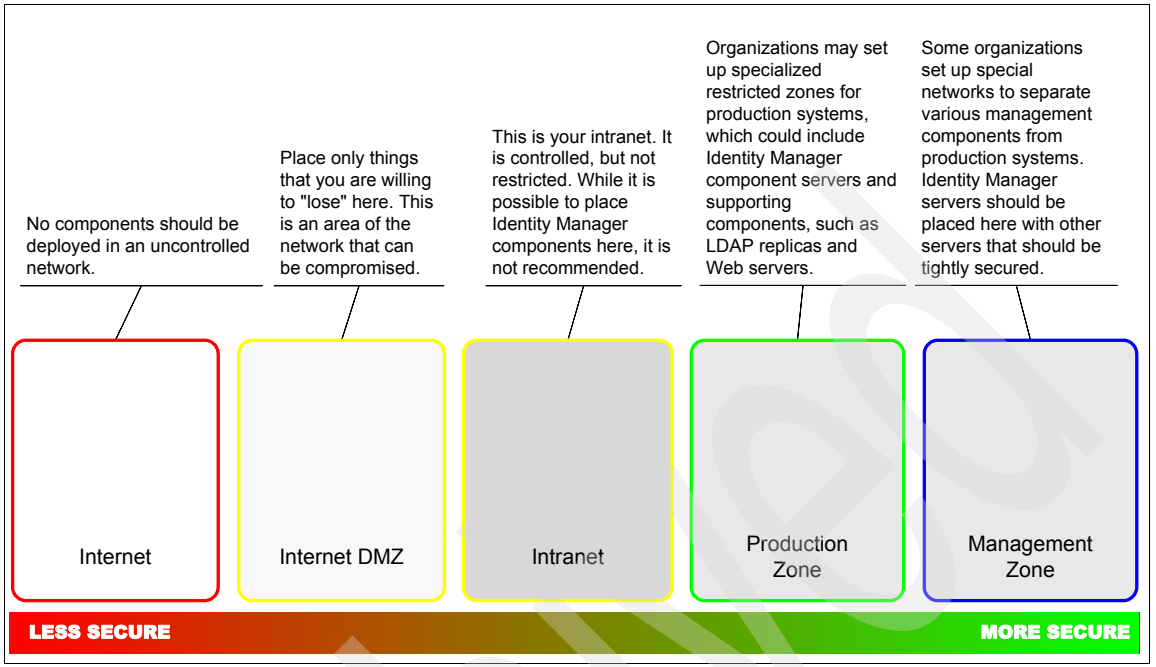


Figure 19-1 Network zones for Identity Manager placement

Because all Identity Manager components operate on information that should be tightly secured we recommend that these components all be placed in a restricted/management zone, except for the components that communicate with other non-Identity Manager components. So one or more Web servers and adapters may be deployed in the Production Zone and one or more Web servers may be deployed in the DMZ to manage external requests from business partners or customers if no general access control solution, such as Access Manager WebSEAL is in place.

Remember, these are suggestions based on the common security architecture subsystems and network models found in Chapter 2, “Common security architecture and network models” on page 19. There are many models that may be constructed. This is meant to be *best practice*. Walkthroughs of complete business processes, including exceptions, can help you create a viable solution and refine the requirements.

19.2 An Identity Manager scenario

For this scenario, we revisit Areally Big Investment Corporation, first introduced in 16.1, “Company profile” on page 492.

Areally Big Investment Corp. has been aware for some time that managing their information infrastructure more efficiently could produce a significant cost savings. After examining operational costs and conducting a cost analysis study, the company established that the cost savings are great enough to warrant an identity management project. They have also established that IBM Tivoli Identity Manager is the solution that will best fit their needs.

19.2.1 Business requirements

To reduce overall IT operational costs and to centralize user management, Areally Big Investment Corp. established that implementing an IBM Tivoli Identity Manager solution will mitigate security risks to an extent that the residual risk is acceptable to the business. They have also established that the return on investment is acceptable in the time frame needed to implement the solution.

Because of their worldwide presence, several languages will be supported. Also, some help desk tasks and administration tasks need to be delegated to their regional facilities.

The corporate vision is to continue to increase employee productivity and prevent customers from becoming dissatisfied, while reducing overall costs of operations. During the cost analysis, the need to apply uniform security policies across platforms was also identified. To simplify the process, the CEO has prefaced the requirements with three words: *security, efficiency, productivity.*

Keeping the three-word-directive in mind, we further refine the requirements into:

- Unified account management
- Single authoritative data source
- Simplified sign-on and unified user experience
- All administrative operations related to user and account management, including creation, modification, suspension, and password reset, need to be executed correctly and in a timely manner.
- Reduce the costs of administering users and their accounts. The CEO is keen to gain cost savings by reducing the amount of work the administrators have to do. The areas identified where savings could be made include:
  - The effort required to reset passwords for users who have forgotten theirs.
  - The effort required to manually create accounts when a person joins the company.

  - The effort required to add new accounts (and remove old accounts) when an employee changes job roles.
- The corporate security policy should be enforced for all user accounts and their attributes, access rights, and password rules. User accounts inconsistent with the policy should generally not be allowed.
- The identity management solution must not be so rigid that it prevents the bank from responding to emergencies and temporary exceptional needs. Administrators must be able to override the system's defaults and policies when necessary.
- The user and account management historical data has to be available from a corporate-wide perspective in order to verify whether the system works according to the guidelines and policies. These logs can help understand shortcomings and implement future improvements.
- Ease of compliance with regulations and audit requirements. Even the bank has put a big effort in this, they fail on external audits. The solution needs to address these problems:
  - Many employees have access to systems that they should not because they:
    - Changed job roles and retained access from their old job role.
    - They are friends with the system administrators and were granted special access without any form of independent check or review of the request.
    - They have left the company, but their accounts have not been deleted.
  - There is no reporting available to verify security compliance.
  - There is no periodic certification of users' access rights.
- The identity management solution must be implemented in a secure manner. It must ensure that:
  - Sensitive data is protected from unauthorized access.
  - Audit data is protected from unauthorized alteration.
  - The system is protected from unauthorized users.
- The identity management solution must be multilingual. Its user interface, reports, and e-mail notifications and documentation must be available in the user language.

19.2.2 Functional requirements

We extract functional requirements by mapping business requirements to their underlying reasons. We expand the reasons in increasing detail until we find

problems that can be solved using capabilities of Identity Manager. Our functional requirements will tie these low level reasons for a business requirement to the Identity Manager capability that will fulfill that business requirement.

Let us examine business requirements, and search for reasons and the functional requirements.

- Business requirement: Unified account management.

  There is a different interface to every platform and system the bank has. Each one has a way to manage its specific accounts. With so many platforms and systems, Areally Big Investment Corp. wastes a lot of resources to manage its accounts. This can be solved by the integration of every platform and system into the Identity Manager solution.

  This leads to the following functional requirements:

  Table 19-1 Functional requirements for unified account management

Requirement	Description
1	All platforms and systems accounts will be managed centrally.

- Business requirement: Single authoritative data source.

  Areally Big Investment Corp. has several user data repositories. Even if this data was eventually pulled from the HR System or a similar system for business partners, several paths were followed and now they do not reflect the same data. There is a need of a single authoritative data source that will be the only source for all platform account attributes.

  This leads to the following functional requirements:

  Table 19-2 Functional requirements for unified account management

Requirement	Description
1	All platforms and systems accounts will be managed centrally.
2	All identity data will be pulled from a central point.

- Business requirement: Simplified sign-on and unified user experience.

  After applying a new security policy for passwords, users have to change passwords more frequently than before. This leads to users forgetting their passwords more often, which results in many password reset requests. Users are less likely to forget their passwords if they use the same password for all of their accounts.

  Password management should be done in a common interface, whatever accounts users have.

This leads to the first two functional requirements shown in Table 19-3.

Table 19-3 Functional requirements for timely password management

Requirement	Description
3	Users will have a single password for all of their accounts.
4	A common Web interface will be used for password management.

- Business requirement: Identity management should be executed quickly and correctly.

  There are two main problems in this area: system administrators are unable to keep up with the volume of requests, and approvals are not being processed in a timely manner.

  The biggest burden on administrators is the increasing number of password reset requests. We can reduce the burden on system administrators by delegating the ability to do password resets. This may be done by users' managers, or possibly by the users themselves. This leads to the first two functional requirements shown in Table 19-4.

Table 19-4 Functional requirements for timely password management

Requirement	Description
3	Users will have a single password for all of their accounts.
5	Password resets will be delegated to users other than the system administrators; possibly to the end users.

Another reason that system administrators have trouble keeping up with the rate of requests is that user and account management operations are time consuming and skill intensive. Administrators must waste time manually entering data that could be computed automatically. This is not only time consuming, it is also error prone. This leads to administrators taking more time to repeat requests that were done incorrectly.

Administrators must also learn different management interfaces for each type of account. Administrative productivity could be enhanced by utilizing a common interface to manage different types of accounts centrally.

This leads to the next set of functional requirements show in Table 19-5.

Table 19-5 Functional requirements for timely account management

Requirement	Description
6	Common values are entered automatically.
7	Manually entered values can be checked for correctness.

Requirement	Description
8	Provide a common user interface for administration.

The other major cause of delays in user and account management is the request approval process. Areally Big Investment Corp. identified the following three primary causes for delays in granting approvals:

– An approver may not be available at the time of a request. Requests should not be delayed because an approver is out of the office. Approvers should be able to delegate their responsibilities if they know they will be unavailable.

– Approvers may be too busy or receive too many requests to respond quickly. Approvals should be assigned to teams instead of to individuals. It must be possible for the team members to assign and take ownership of individual approval requests.

– Approvers may forget that they are responsible for a request. An approver who doesn't act on a request must be periodically reminded that the request is waiting. If they still don't respond, the request should be escalated to a different approver.

These issues are addressed by the next set of functional requirements shown in Table 19-6.

Table 19-6 Functional requirements for timely request approval

Requirement	Description
9	Allow delegation of approval responsibilities.
10	Support collaboration by multiple approvers.
11	Remind approvers of waiting requests.
12	Escalate ignored requests.

► Business requirement: Reduce administrative costs.

Areally Big Investment Corp. identified three areas in which they want to reduce the costs associated with user and account administration: password resets, account creation for new employees, and account maintenance for users who change job roles. These three tasks occupy much of the time of many high paid system administrators. We can reduce the number of administrators, and allow the remaining administrators to focus on higher value projects, if these tasks can be automated or delegated to other users.

Password resets have already been discussed in the context of the business requirement to execute requests quickly and correctly. Functional requirement

B (delegation of password resets) will also satisfy the cost reduction business requirement.

Areally Big Investment Corporation's administrators are responsible for creating new accounts for newly hired employees. Some of these accounts, such as e-mail and Windows access, are common to all employees and use similar settings on all accounts. The automation of the setup of these accounts would allow system administrators to concentrate on more useful work.

The system administrators are also responsible for creating new accounts and suspending existing accounts when employees change job roles. Many job roles have a standard set of accounts and access rights that must be given to a user when they enter the role, and must be removed when the user leaves the role. This is another case where automation could relieve the administrators from repetitive tasks.

The functional requirements for cost reduction are shown in Table 19-7.

Table 19-7 Functional requirements for cost reduction

Requirement	Description
5	Password resets will be delegated to users other than the system administrators; possibly to the end users.
13	Automatically create common accounts when a person is employed.
14	Automatically add and remove accounts and access rights when a user changes job role.

- Business requirement: The corporate security policy should be enforced for all user accounts.

  Accounts sometimes have attributes that do not comply with the corporate security policy. This may be accidental due to mistakes made by administrators or ignorance of the violated policies. Some non-compliant accounts may also be the result of intentional misconduct by administrators. These may be cases of administrators who are too lazy to follow the policy, or the administrators may have malicious reasons for violating the policies. In either case, there is no verification of the values entered by system administrators when they are creating and modifying accounts.

  Violations of the corporate security policies can be reduced by setting the values of account attributes automatically, when possible. Further reductions in violations can achieved by introducing compliance checking on attribute values that are set manually. Both of these strategies rely on having a centralized user interface for account management, and a way to find changes made to accounts outside of the central user interface.

There is substantial overlap between the functional requirements for insuring compliance with security policies, and the functional requirements for timely account management (shown in Table 19-5 on page 579). The requirements for a common user interface, automatic calculation of common account attribute values, and checking of manually entered values, all help to both make system administrators more productive, and to enforce compliance with the security policies. In addition there are requirements that the security policies will still be enforced even if an account is changed outside of the centralized account management tool, and if the policies themselves are changed. The combined functional requirements for compliance with security policies are shown in Table 19-8.

Table 19-8 Functional requirements for compliance with security policies

Requirement	Description
6	Common values are entered automatically.
7	Manually entered values can be checked for correctness.
8	Provide a common user interface for administration.
14	Account changes made outside of the common interface are detected an checked against the security policies.
15	Changes to security policies are checked against existing accounts.

- Business requirement: Enforcement of security policies must be flexible enough to allow for emergencies and exceptions.

  Areally Big Investment Corp. realizes that there will always be cases where an exception to a security policy will be needed. No set of policies will ever be able to foresee every combination of account attributes that might be needed by a user. When temporary or emergency needs arise, there must be a way that the administrators can override the security policies.

  Areally Big Investment Corp. anticipates two likely scenarios where exceptions to the security policies will be needed. The first is when a user needs temporary administrative rights in order to perform software installation or maintenance. The second situation when this need will arise is when a user changes their job role. The security policy may require that the person lose some access rights when they leave their old job role. But such changes in responsibilities are rarely instantaneous. A user who is changing departments will often go through a transition period during which they will need the access rights of both their new and old job roles. It must be possible to detect accounts that are out of policy, and have a designated administrator define how long the account may remain out of policy before it is brought into compliance automatically.

The functional requirements for flexibility in security policy enforcement are listed in Table 19-9.

Table 19-9 Functional requirements for flexible compliance with security policies

Requirement	Description
16	An administrator can create or change an account even if the resulting account violates the corporate security policies.
17	Designated administrators will be notified when non-compliant accounts are detected.
18	The designated administrators can decide how long the account may remain non-compliant. After this period expires the account will be automatically brought into compliance with the security policies.

- Business requirement: User and account management historical data has to be available for verification and future improvements.

  In the current system, account information is scattered all over the corporate systems. It is not easy to understand how many user accounts are being used in the enterprise, at what rate they are growing, and when the system should be expanded due to increasing account numbers, and so on. The information is indispensable for verifying the current system and for making future plans to expand it. A central logging system can provide this information. This requirement is shown in Table 19-10.

  Table 19-10 Functional requirements for availability of historical data

Requirement	Description
19	A central logging system is needed.

- Business requirement: Improve audit compliance.

  Areally Big Investment Corp. wants to improve their audit compliance in the following three areas:

  - Requiring users or their managers to periodically certify the users' continuing need for their accounts and access rights.
  - Removal of accounts or access rights that are no longer needed. This may be divided into three different populations of accounts:
    - Accounts belonging to users who have left the company.
    - Accounts belonging to users who have changed job roles.
    - Accounts that were not certified as still needed.
  - Reporting capabilities for finding accounts that are in violation of the corporate security policies.

Requiring certification of need for accesses is the best way to prevent temporary accesses from becoming forgotten accesses. Areally Big Investment Corp. is concerned that users who are given temporary access to an application or some data will keep that access even when the access is no longer needed. It's reasonable for people to do this if they aren't certain that they are finished with their work that requires the access. The problem is that people will eventually forget that they have the access, and will never request that it be removed. At worst, their unused accounts or access rights are left for hackers to find. At best, determining who had access to some data or an application becomes more difficult.

Removing obsolete accounts and access rights has obvious benefits for audit compliance. The functional requirements for this area will have some overlap with the functional requirements for flexible security policy enforcement shown in Table 19-9 on page 583. The functional requirements that administrators be notified of non-compliant accounts, and that the accounts be brought into compliance at some point in time, help to satisfy both the policy enforcement and audit compliance business requirements. This will meet the need to remove accounts and access rights that result from a user changing job roles.

Removing accounts belonging to people who leave the company requires that Identity Manager receives regular updates from one or more authoritative sources of identity data. This data must be updated in a timely manner so Identity Manager can disable the accounts of former employees without excessive delays.

The functional requirements for improved audit compliance are listed in Table 19-11.

Table 19-11 Functional requirements for improved audit compliance

Requirement	Description
20	Designated administrators will be notified when non-compliant accounts are detected.
21	The designated administrators can decide how long the account may remain non-compliant. After this period expires the account will be automatically brought into compliance with the security policies.
22	Account owners and/or their managers will be periodically asked to certify their continuing need for their accounts and access rights.
23	Accounts and access rights that are not certified will be disabled or removed.
24	A regular feed of identity data from authoritative Areally Big Investment Corp sources into Identity Manager will be established.

Requirement	Description
25	An employee's accounts will be disabled or removed when the identity feed shows that an employee has become inactive.
26	A reporting mechanism will be available that identifies accounts that are not in compliance with the corporate security policies.

- Business requirement: The identity management solution must be secure.

  A poorly designed identity management solution poses a security risk. There are three primary areas of concern:

  – Confidentiality of sensitive data

    Identity Manager stores sensitive data in its data stores. It also transmits sensitive data between its individual components. The stored data and the data in transit must be protected from unauthorized access.

  – Integrity of audit data

    Identity Manager administrators have a great deal of power. By manipulating provisioning policies they could create accounts with almost any rights they want on any platform controlled by Identity Manager. Since it is difficult to prevent an administrator from abusing their powers, it's important that an audit trail be maintained of the administrator's actions. The administrators who are being monitored with this audit data must not have the ability to manipulate the audit data.

  – Authentication of system users and components

    Identity Manager must be protected from access by unauthenticated or unauthorized users. Each Identity Manager component must also authenticate the other components with which it communicates.

  The functional requirements for the security of the identity management solution are shown in Table 19-12.

Table 19-12 Functional requirements for application security

Requirement	Description
27	Stored sensitive data will be protected from unauthorized access.
28	Transmitted sensitive data will be protected from unauthorized access.
29	The actions of Identity Manager users and administrators will be tracked in an audit trail.
30	Identity Manager administrators will not be able to manipulate the audit data or settings.

Requirement	Description
31	Identity Manager components will be protected from access by unauthenticated or unauthorized users.

- Business requirement: The identity management solution must support English and Spanish speaking users.

  Areally Big Investment Corp. wants the employees of the Mexico offices to be able to access the identity management solution in their native language. This is important so that these employees will accurately understand the actions they are performing with the system. The functional requirements for this area are shown in Table 19-13.

Table 19-13 Functional requirements for national language support

Requirement	Description
32	All displays, notifications, and online documentation of the identity management solution must be available in both English and Spanish.

Note: Some functional requirements overlap or are addressed by the Tivoli Access Manager for Enterprise Single Sign-On implementation. Anyway, Areally Big Investment Corporation decided to implement Tivoli Identity Manager as their complete Identity Management solution and keep Tivoli Access Manager for Enterprise Single Sign-On for single sign-on purposes.

19.2.3 Designing the solution

Given the mandate of security, efficiency, and productivity, and the 19.2.2, “Functional requirements” on page 577, developing functional requirements for the solution is the first place to begin. Creating a questionnaire that can be administered in individual interviews or workshops is helpful for uncovering the many contingencies to implement the solution. The place to begin is the lifecycle management of an identity. How is an identity created? Who initiates the process? Who maintains information up-to-date? What operating systems are in play? How many systems will the user have access to? Who actually completes the work? Is there an audit trail? Other areas that require focus include passwords, group membership, organizational role membership, managed systems and applications, policies, and workflows.

After the business and functional requirements are defined you can look at the security design objectives. The security objectives and the associated subsystems become the basis for the conceptual architecture and the implementation phase. The security phase should include identity management,

password management, policy management, business process management, and audit management, as well as all standards, guidelines, and policies that relate to operations.

Also, a review is needed of the existing job roles in the organization including organizational structure, departments, and teams. This review should also include the necessary system access requirements, attributes, applications, and so on. Figure 19-2 gives an idea of how the process should begin to come together.

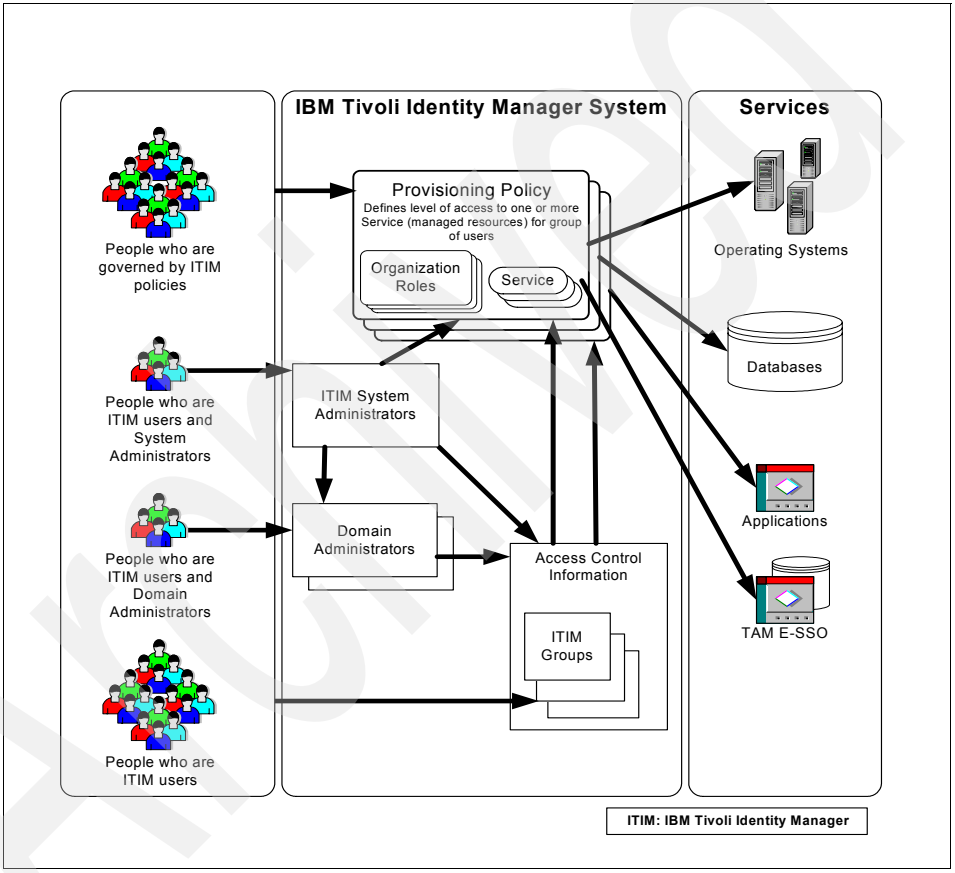


Figure 19-2 IBM Tivoli Identity Manager relationships

How to collect the criteria and in what order to collect them will be defined by the project or by utilizing a specific security architecture methodology. For this scenario, we use very basic requirements for identity management.

Areally Big Investment Corporation set the criteria for the implementation of IBM Tivoli Identity Manager. The goal is implement the features that give the most valuable benefits, so the first implemented features are:

- Single authoritative data source
- Unified account management
- Simplified sign-on

Management has also stated that the ability to delegate administration tasks based on security models is a requirement of the project, as well as having a common user-friendly interface that is intuitive when changing passwords or completing administrative tasks. They also want to integrate the business process into the identity system with clear auditability of user activity and resource usage, have a central location to manage all identities for all systems within the company in one authoritative data source, and have the ability to universally apply security policies to user accounts.

Given those requirements, a basic IBM Tivoli Identity Manager physical architecture begins to take shape. Figure 19-3 on page 589 shows a sample basic Identity Manager architecture, given the infrastructure we currently have and the requirements we need to meet.

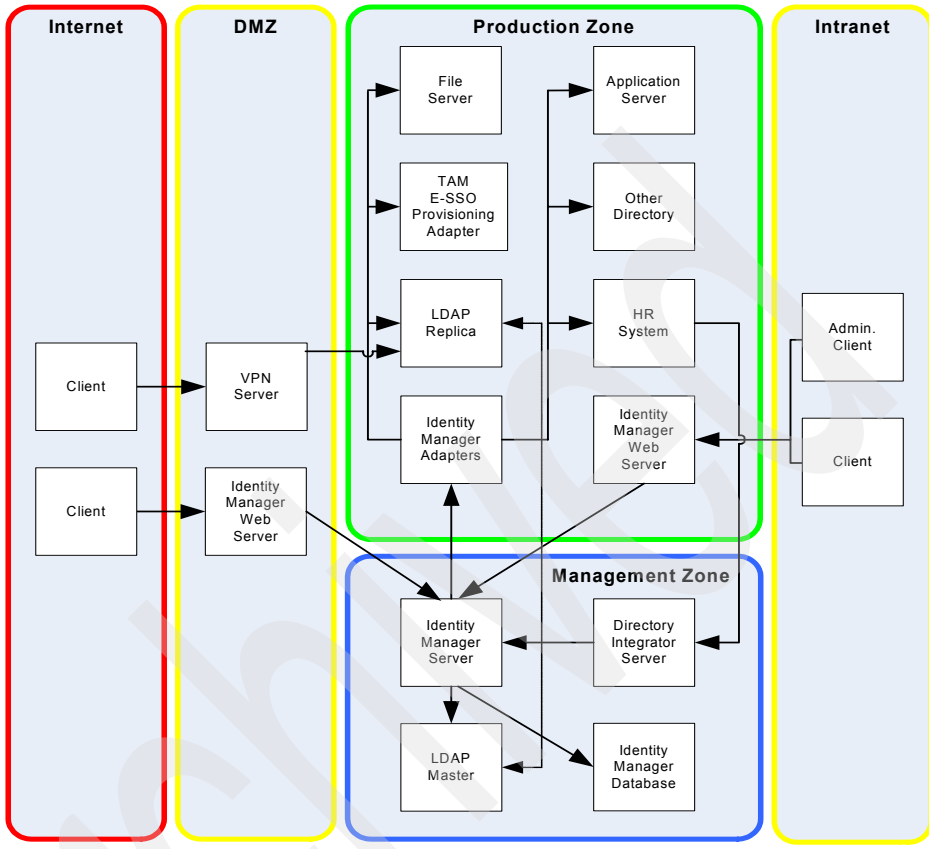


Figure 19-3 Basic physical architecture for Areally Big Investment Corporation

The IBM Tivoli Identity Manager server components are all located within the management zone. Access to these systems is restricted and controlled. However, the Identity Manager adapters reside in the production zone and also the Web server for intranet users. There is also one Web server for internet users that reside in DMZ.

Note the following:

- The authoritative source for identities is the HR system.
- HR system users are managed by IBM Tivoli Identity Manager.
- After communicating the necessary identity information to the adapters, access for users is granted or denied by the resource itself, and not by the Identity Manager adapter.

Areally Big Investment Corporation decides to maintain Identity Manager components in the main data center. The fact that Tivoli Identity Manager is a Web application helps a lot with this. There is, however, one exception to this, that is the deployment of an Identity Manager adapter server at each production zone of a foreign data center.

Because of local regulations, each country has its own HR system. There is also some contractor systems that are authoritative sources for identities of non-employees. For each identity authoritative source there will be one process that pulls identity data and push it to Tivoli Identity Manager.

Figure 19-4 shows the scenario for a sample worldwide implementation.

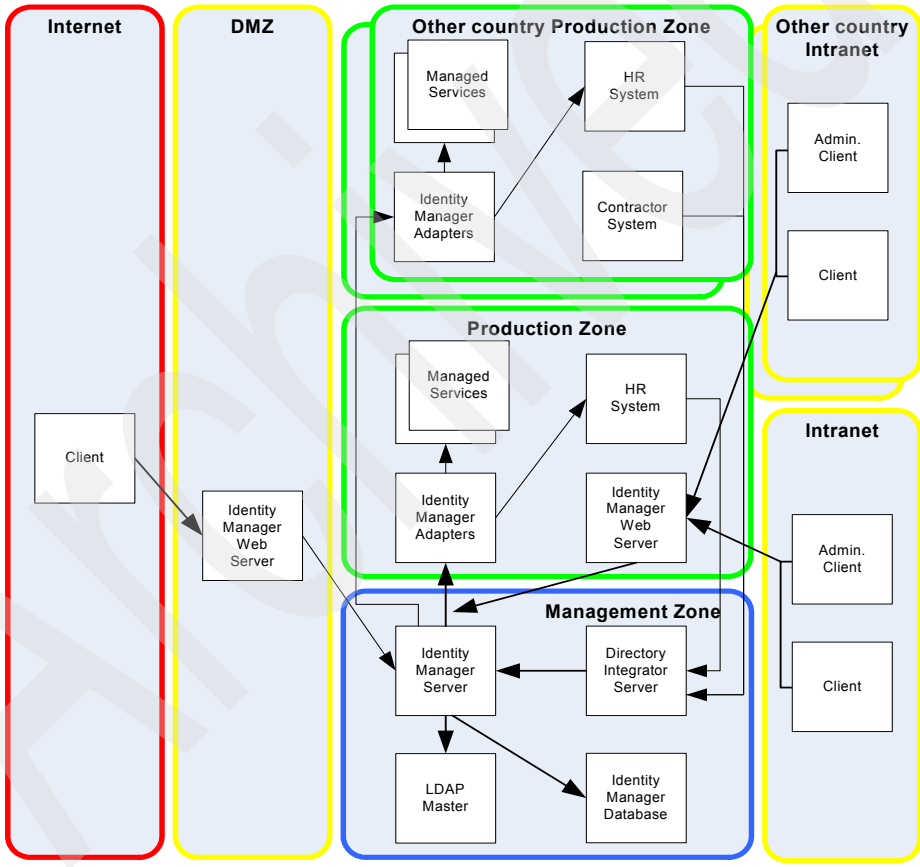


Figure 19-4 Worldwide implementation

19.3 Tivoli Access Manager for Enterprise Single Sign-On Provisioning Adapter

Because of our previous Access Manager for Enterprise Single Sign-On deployment, discussed in Chapter 16, “Tivoli Access Manager for Enterprise Single Sign-On scenario” on page 491, Areally Big Investment Corporation decided to automatically provision new users’ credentials along with their Tivoli Identity Manager account provisioning process.

This can be done by implementing the Access Manager for Enterprise Single Sign-On Provisioning Adapter, as introduced in 15.1.10, “Provisioning Adapter” on page 472.

In this section, we focus on the Tivoli Identity Manager specific working details of the Access Manager for Enterprise Single Sign-On Provisioning Adapter.

The integration is achieved through an update to the Access Manager for Enterprise Single Sign-On credential repository for every successful provisioning of account credentials done by Tivoli Identity Manager.

To accomplish this, the Access Manager for Enterprise Single Sign-On Provisioning Adapter integration with Tivoli Identity Manager uses custom Tivoli Identity Manager Entity Operations called *Tivoli Identity Manager Workflow Extensions*. These *workflow extensions* call the Access Manager for Enterprise Single Sign-On Provisioning Adapter itself. From a Tivoli Identity Manager perspective, the Access Manager for Enterprise Single Sign-On user credentials are not considered *normal accounts* but *shadow accounts*, which refer to a provisioned credential that is not actually stored as an object in the Tivoli Identity Manager account’s container. Tivoli Identity Manager does not store any Access Manager for Enterprise Single Sign-On specific data, and the communications only occur from Tivoli Identity Manager to the Access Manager for Enterprise Single Sign-On Provisioning Adapter, never the other way around.

Figure 19-5 on page 592 shows the relationship between the Tivoli Identity Manager server, Access Manager for Enterprise Single Sign-On Provisioning Adapter Extension Classes, and Access Manager for Enterprise Single Sign-On Provisioning Adapter through a change password Tivoli Identity Manager operation. In this case the following occurs:

1. The *change password* operation starts.
2. The Tivoli Identity Manager Change Password Extension creates the account password change request to a particular service, in this example, an Active Directory.

3. After successful password change, the Access Manager for Enterprise Single Sign-On Provisioning Adapter Extension is executed. It provisions the same password already updated in the previous item.
4. The Access Manager for Enterprise Single Sign-On Provisioning Adapter Extension initiates a request to the Access Manager for Enterprise Single Sign-On Provisioning Adapter, which updates the Access Manager for Enterprise Single Sign-On credentials for that specific user.

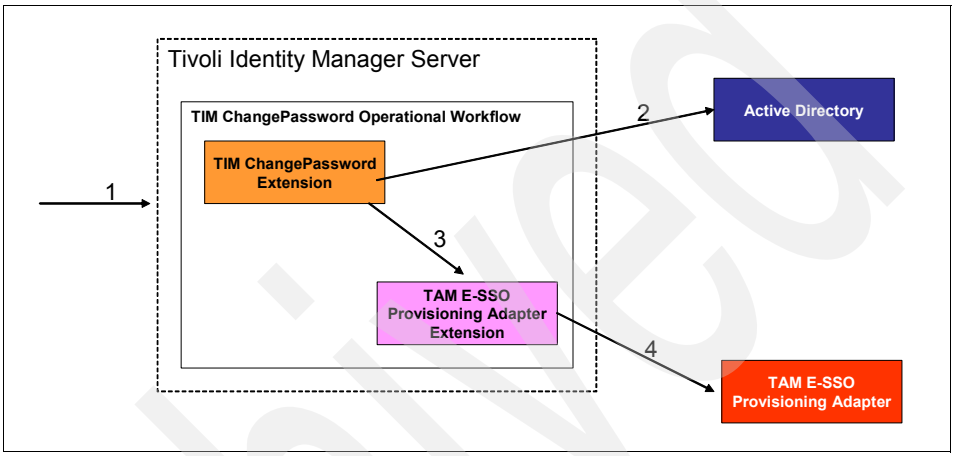


Figure 19-5 Identity Manager server and Access Manager for Enterprise Single Sign-On Provisioning Adapter relationship

Updates to the Access Manager for Enterprise Single Sign-On repository only occurs after the update to the native repository is successful. Any failed Tivoli Identity Manager account operation skips the Access Manager for Enterprise Single Sign-On specific workflow extensions.

19.4 Tivoli Identity Manager high-availability

This discussion is about the high availability aspects of the Identity Manager software components. For each component, the concepts of a fault tolerant hardware configuration and high availability operating system based infrastructure should be considered and evaluated and the costs weighed up against the risks. These aspects however, are more generic systems design concepts common to most projects and specific to an organization's operational environment. They are not necessarily specific to Identity Manager and hence are not within the scope of this discussion. In most cases a combination of the various approaches will be used, but no two environments will typically use a standard uniform approach due to the unique constraints, dependencies and

priorities of each and should be evaluated and planned for in by the project team in consultation with stakeholders and system owners.

The software components relevant to an Identity Manager deployment when considering a high availability solution are as follows:

- Application server
- Directory server
- Relational Database
- Identity Manager adapters
- Identity Manager Reverse Password Synchronization components

19.4.1 Application server high availability

The application server runs the Identity Manager application that performs all the business related operations and provides the Web interface to users. The application server used by Identity Manager is WebSphere Application Server, which provides the ability to run as a WebSphere Application Server cluster using the IBM WebSphere Application Server Network Deployment.

In IBM WebSphere Application Server Network Deployment, the distributed administrative work is accomplished by the Node Agent server that resides on each node and the Deployment Manager that acts as a central point for administrative tasks.

The Node Agent server and the Deployment Manager server both use a repository of XML files on their own nodes. The master repository data is stored on the Deployment Manager node. That data is then replicated to each node in the Administrative domain (or cell). The Deployment Manager server and the Node Agent servers keep these files synchronized. The synchronization process is unidirectional from the Deployment Manager to the Node Agents to ensure repository integrity.

However, the first point of failure is the Web server. A network load balancer device need to be deployed to balance https connections for two or more Web servers. The Web servers communicate to WebSphere Application Server through a plug-in that already has the capability to talk to a WebSphere Application Server Cluster.

Figure 19-6 shows a typical deployment of these components.

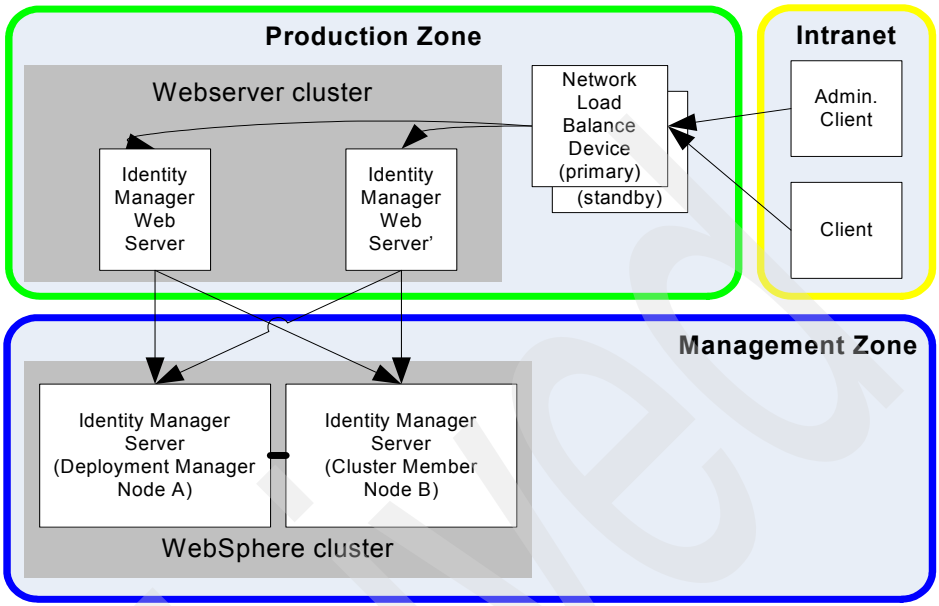


Figure 19-6 WebSphere Application Server Cluster

Using the native cluster will allow the Identity Manager application to run as a clustered application and leverage the benefits that are provided with that, for example, session failover in the event of a cluster member being unavailable, and sharing the load between all cluster members.

For further details, refer to *IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series*, SG24-6198.

Note: It is a best practice to leverage the functionality available in a security reverse proxy component such as Tivoli Access Manager WebSEAL to perform the authentication and authorization for users into Identity Manager. WebSEAL will also automatically perform the load balancing and failover aspects in the event of an application server instance failure.

19.4.2 Directory server high availability

Identity Manager requires an LDAP server to store essential data such as users, accounts, policies and so on. As a result, it is an extremely critical component. Most LDAP servers have some level of functionality to allow for a high availability deployment. This discussion is based on Tivoli Directory Server Version 6.

IBM Tivoli Directory Server allows for multiple LDAP servers to be configured with replication between them to ensure data integrity is maintained. Each Tivoli Directory Server server can be configured as a read/write enabled server or as a read-only replica. However, a read-only replica has low value for Identity Manager. Most Identity Manager operations issues LDAP write operations. Even a login process in Tivoli Identity Manager issue a write operation at the Directory Server. So a high-available read/write directory is the goal of this discussion.

Identity Manager allows for configuration against a single “logical” LDAP. This means that it needs to refer to a URI (Uniform Resource Identifier) that will allow access to an LDAP server. Basically what is needed to achieve high availability is to make this URI high available. The Tivoli Directory Server Proxy is a great solution to make this logical URI point to a high-available Directory Server infrastructure.

To eliminate any single point of failure all components are at least duplicated. To maintain integrity, an NTP server (network time protocol) or any other time synchronization process should be used, specially when multiple read/write directory servers are being used.

Figure 19-7 shows a typical deployment of these components.

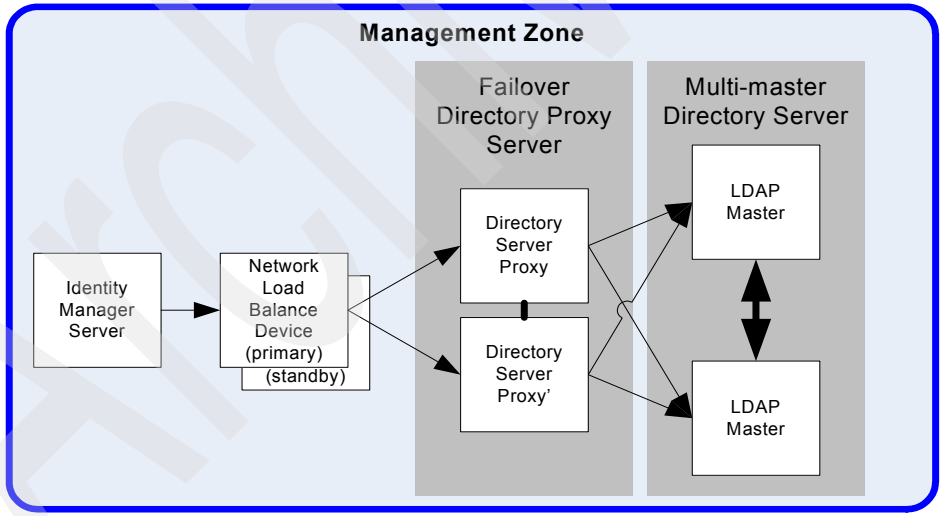


Figure 19-7 Directory Server high availability

19.4.3 Relational database high availability

Some potential relational database high availability scenarios for Identity Manager are similar to the scenarios detailed in 19.4.2, “Directory server high

availability" on page 594. The concepts are similar and can be extrapolated from the LDAP discussion.

However, the way a LDAP directory server is implemented simplify a lot the effort to deploy a high availability scenario. Normally the high availability solutions available for common relational database products (such as IBM DB2) are more sophisticated than for LDAP products. Relational database high availability strategies need generally to be more tightly coupled with high availability operating system configuration options, systems management (automation, virtualization), and storage solutions (for example, Storage Area Networks) to achieve the desired results. There are prescribed methods to deploy a high availability relational database within specific product documentation.

Taking IBM DB2 UDB version 8.2 as an example and using a combined approach leveraging operating system high availability features, the solution design team may choose to deploy DB2 as follows:

- Operating system cluster with DB2 active/standby configuration: In the event of the active DB2 instance failing, the operating system clustering software starts the same instance on another node in the operating system cluster. This requires that all nodes in the operating system cluster have access to the same shared disk. While relatively simple in terms of DB2 clustering, this introduces delays during failover while the new processes are started and any in-flight transactions are rolled back. The database is accessed through the cluster address, so that no change in the Identity Manager database configuration is required during failover.
- DB2 *mutual takeover multiple partition* configuration: All nodes in the database cluster operate in parallel. The database is partitioned so that if any server in the cluster fails, its partitions are failed over to the remaining nodes in the cluster. As with other strategies, there are various considerations in using this approach. The configuration still requires time for the "failed-over" partitions to be recovered although as each partition has less than the whole volume of data it is generally faster than an active/standby configuration. There needs to be database analysis performed to determine an appropriate database schema required for constructing a partitioned version of the Identity Manager database. This approach also requires that all servers have access to the file systems containing the database and transaction logs. The database is accessed through the cluster address, so that no change in the Identity Manager database configuration is required during failover.
- *DB2 High Availability Disaster Recovery* (HADR): This solution involves using the DB2 automatic log shipping functionality to a secondary stand-by server which applies the logs as it receives them. If the primary active server fails then the DB2 client is automatically re-routed to the secondary failover server. As no crash recovery is required, the failover to the secondary can be achieved in a minimal amount of time. Note that only one server can be active

and read and written to by a client. The secondary stand-by servers cannot participate in reads. There also needs to be careful planning if multiple failover stand-by databases are to be receiving updates. The DB2 client takes care of the failover hence there is no need for manual intervention in the Identity Manager database configuration during failover. Figure 19-8 shows a typical deployment of these components.

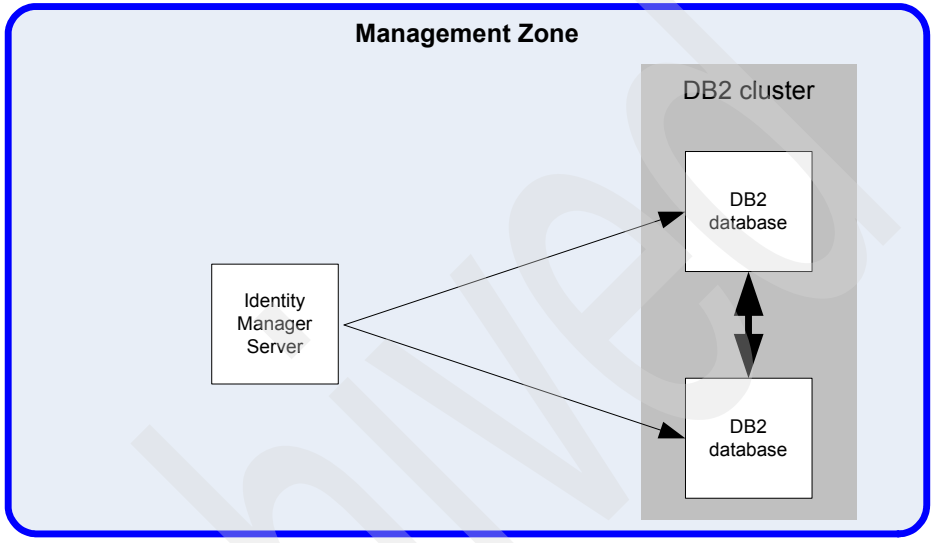


Figure 19-8 DB2 cluster using HADR

Note: The DB2 options outlined are intended as examples. These are not the exhaustive high availability strategies available with DB2. For specific details and other options, refer to the DB2 8.2 product documentation.

19.4.4 Identity Manager adapters high availability

Identity Manager manages accounts on managed resources through the use of adapters. This section speaks generically about adapters in general. It should be noted that there are different adapters for each distinct type of managed resource but the concepts discussed can be generally applied across the different types of adapters. For example, the concepts apply to the Active Directory adapter as well as the Lotus Notes adapter and so on.

Account operations issued by Identity Manager are executed by the relevant adapter for the type of managed resource the accounts reside on. This includes provisioning, password management and reconciliation operations. It can be argued that account operations are not mission critical and hence do not need to

have high availability requirements factored in for a solution design. In many cases, this may be true. As with many things however, there are exceptions to the rule. It is not our intention to debate the validity of the claims for and against the need for highly available provisioning operations as this can at times become a “religious” discussion. Each deployment has specific requirements and it may be decided that certain operations must be highly available. For example, there may be cases where password resets, account suspensions and account de-provisioning are deemed to be critical operations and must be highly available.

There are two aspects to consider when dealing with high availability for Identity Manager interactions with its adapters and subsequently the adapter interactions with the managed resource hosting the accounts being managed as illustrated in Figure 19-9.

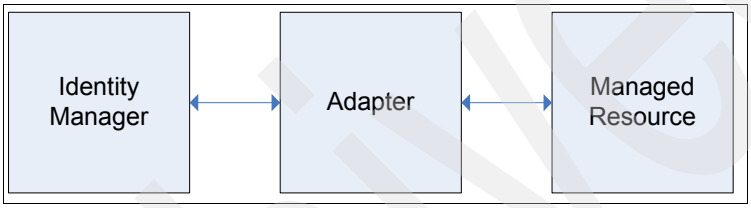


Figure 19-9 Identity Manager interactions with managed resources

The first is the adapter interactions with the managed resource. For example, the Identity Manager Windows Active Directory adapter and its interactions with Windows Active Directory. The high availability aspects between these two components are not within the scope of this discussion as each managed resource has different approaches to high availability and they can vary in completely different ways. For example, a Windows Active Directory environment has a different high availability design and implementation approach compared to a Linux environment, both of which are different to a Tivoli Access Manager environment, and so on. The managed resource is viewed as a logical entity in the context of this discussion and assumed to have been designed for high availability by the solution design team responsible for the managed resource in question. The focus of this discussion is on the Identity Manager specific components required for ensuring account operations are highly available.

Identity Manager, as per the approach taken with the LDAP and database, references its adapters via a URI (Uniform Resource Identifier). As previously mentioned, this is a *logical* location. Given this, consider the following scenarios:

Note: The following scenario considers the use of two adapters in each case. This can be extrapolated to cases where it is determined that there is a need for more than two adapter instances.

This scenario relies on the secondary adapter being available to be used at all times. There is also a requirement to leverage the use of a suitable TCP load balancer. The Identity Manager server is configured to reference the URI of the load balancer which then routes requests to the relevant available adapter as shown in Figure 19-10.

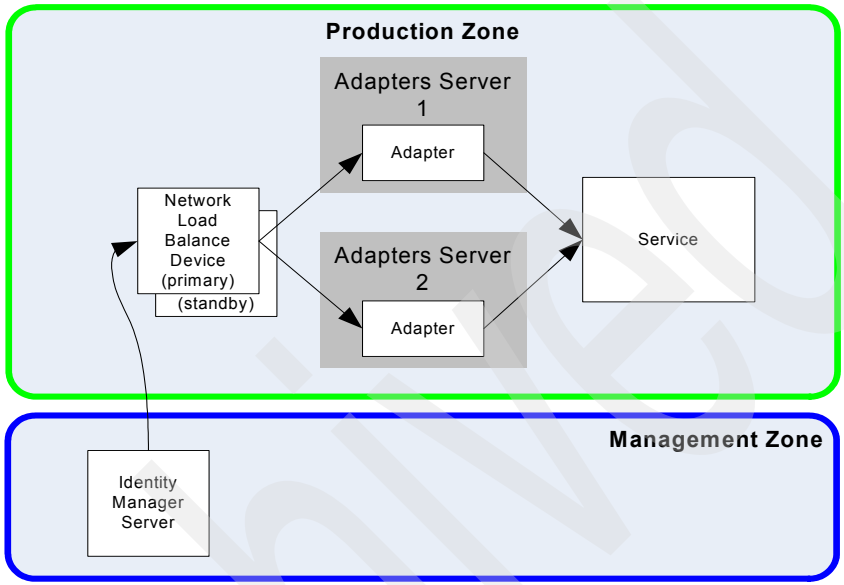


Figure 19-10 Automated failover to secondary adapter

Note that the secondary adapter must be configured exactly the same way the primary adapter is configured and using same encryption keys. If you use a network load balancer that is “http-session-friendly” you should have no problems at all doing load balance. In case of requests that need to run alone (for example a reconciliation), Tivoli Identity Manager would just “lock the service” so any other requests would be hold in the pending queue until the running request finishes, even if there are idle adapters available.

The same scenario can be applied for custom Tivoli Directory Integrator adapters.

Attention: When using Tivoli Identity Manager adapter event notification, make sure that it is enabled in only one adapter instance. This is necessary because there is no mechanism to share the adapter event notification database.

19.4.5 Reverse password synchronization high availability

In 18.3.7, “Lifecycle example” on page 569 we discussed the Identity Manager reverse password synchronization. It can be configured to use the Identity Manager password policies to enforce password rules. This relies on the Identity Manager server being available. The implication of this is that password changes are not possible without the availability of the Identity Manager server as the password policies are required. Note that password rule enforcement by Identity Manager is optional. If not enforced, password changes are possible without the availability of the Identity Manager server. For example, consider a Windows environment where password rule enforcement is enabled. A user presses “Ctrl+Alt+Delete” on the Windows desktop and selects the option to change their password. If the Identity Manager server is unavailable, no amount of attempts will allow them to change their Windows password due to the fact that the new password cannot be verified against the Identity Manager password policies. In the case where password rule enforcement by Identity Manager is not enabled, password changes are successful regardless of the Identity Manager server’s availability. The implication in changing a Windows password without the Identity Manager server being available is that the Windows password is no longer synchronized with the user’s other accounts managed within Identity Manager.

A configuration setting within the Identity Manager reverse password synchronization component specifies the location of the Identity Manager server via a URI. This is typically the physical network location of the Identity Manager server. In a natively clustered Application server environment (such as a WebSphere Application Server cluster), the Identity Manager application is highly available by virtue of the configuration. In the cases where Identity Manager is not deployed into a native application server cluster and high availability is required for reverse password synchronization, there is a requirement to leverage the use of a suitable IP load balancer as detailed in 19.4.2, “Directory server high availability” on page 594. The URI specified within the reverse password synchronization component’s configuration should be the location of the load balancer. There needs to be at least two instances of the Identity Manager application (both using common logical data sources) for the load balancer to leverage to support the requirement for high availability.

Notes:

1. The following scenario considers the use of two Identity Manager application instances. This can be extrapolated to cases where it is determined that there is a need for more than two.
2. While possible, the option to manually reconfigure the Identity Manager reverse password synchronization component to reference the URI of the secondary Identity Manager in the event of the primary Identity Manager failing (hence not requiring the use of a load balancer) is not being considered as in some cases, this requires a restart of the managed resource. This is typically considered unacceptable in cases where the managed resource is a critical piece of corporate infrastructure. For example, a reconfiguration of the Identity Manager reverse password synchronization component for Windows Active Directory requires a restart of the Active Directory domain controller it is installed on.

Load balance Identity Manager application instances

A single reverse password synchronization request made to the Identity Manager server is not typically a long running request. They are generally very short, single communication session events and do not need to manage state between requests hence it is acceptable to use a load balancing/sharing strategy as opposed to being forced into a failover strategy. In the load balancing scenario, both application instances are available and share the reverse password synchronization requests being routed by the load balancer as shown in Figure 19-11 on page 602. Failure of an Identity Manager instance will cause all requests to be routed to the remaining available instance. Identity Manager server unavailability may not be noticeable by the user.

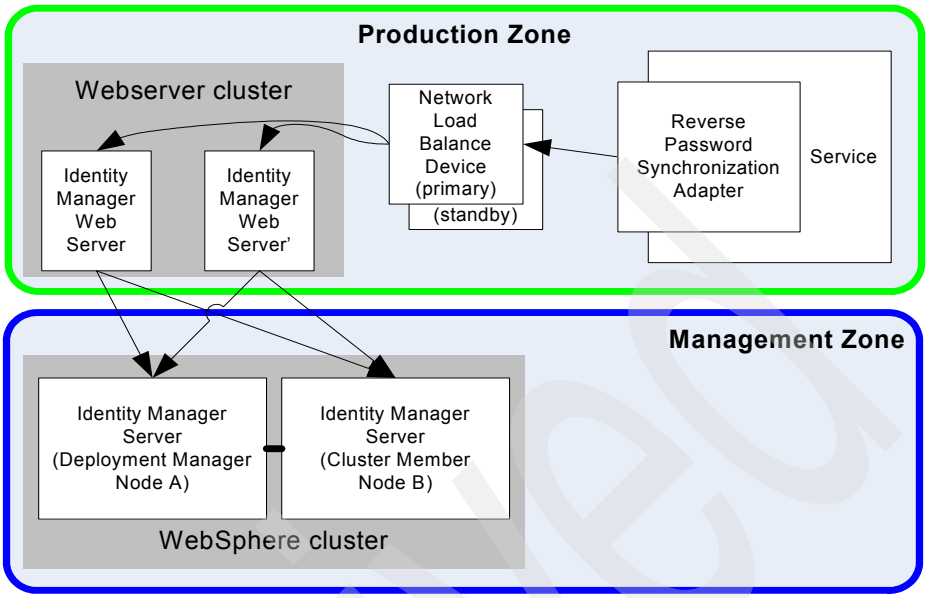


Figure 19-11 Load balance between two Identity Manager server instances

19.4.6 Complete scenario

In the previous sections we have discussed some high-availability scenarios for each component. It is not the goal of this book to discuss every high-availability option, benefits, costs, or the level of availability required. More details about this topic can be obtained in Chapter 8 of the *IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series*, SG24-6198.

Areally Big Investment Corporation has a worldwide Tivoli Identity Manager implementation. They decided to implement a high-level availability within two separate sites. Figure 19-12 on page 603 shows both sites and its components.

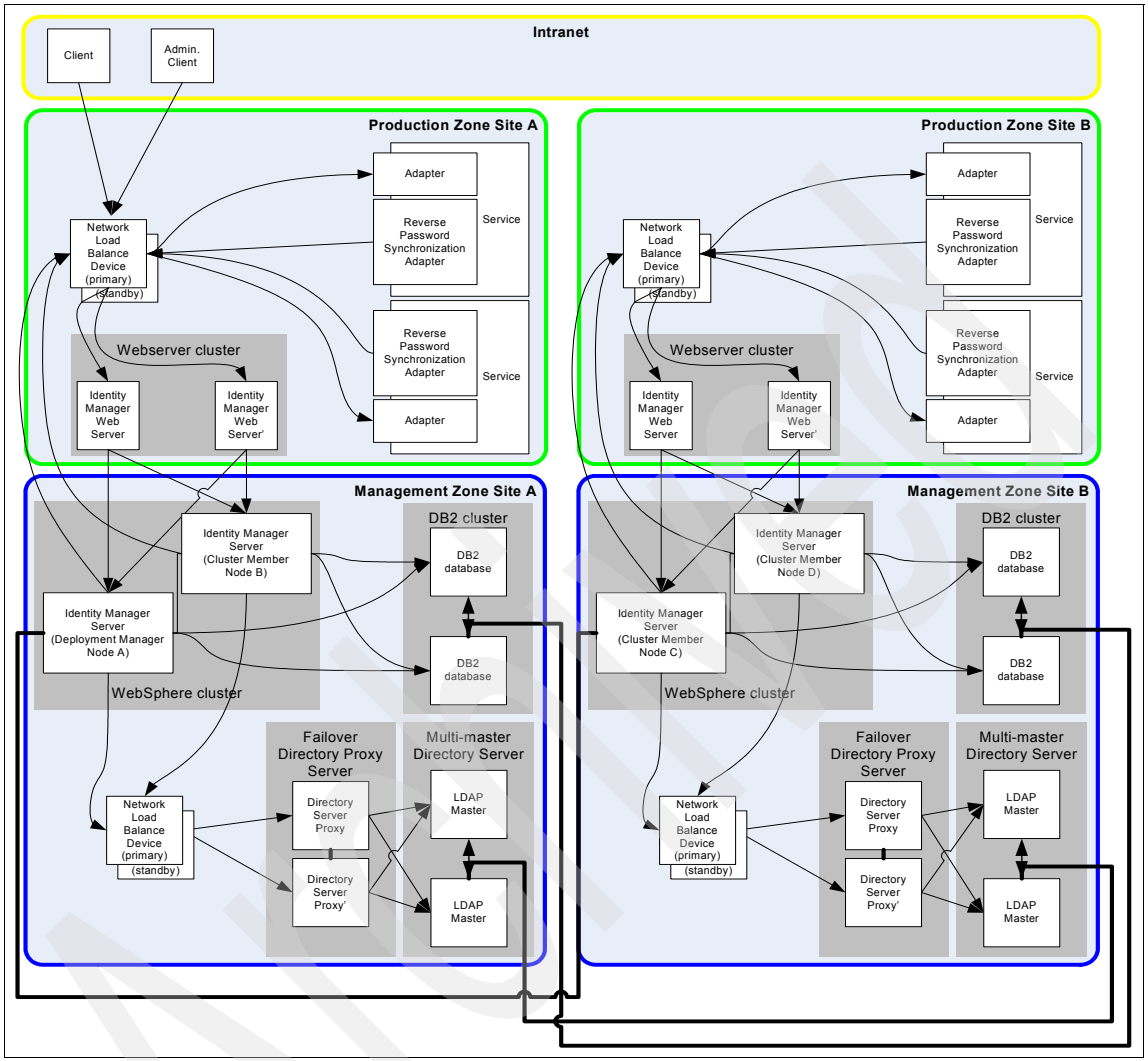


Figure 19-12 Complete physical component distribution

19.5 Importing and synchronizing user data

Identity Manager is designed to be a central location for corporate identity management. Because Identity Manager requires its own user registry and cannot share the user objects that are in the user registry of another application (such as Access Manager or a corporate directory), you have to create new user

records in Identity Manager or import existing user data records from other data resources.

For Areally Big Investment Corp, there is more than one authoritative source for identities:

- One HR system for every country
- Some countries also have a contractor system for non-employee persons

The process of importing and synchronizing identity data from this systems to Tivoli Identity Manager is called *identity data feed*. This data will be available to managed services through the Identity Manager solution. More than identities and it's attributes, the integration of business process into Identity Manager relies on the identity datafeed, so it is one of the most important pieces of the identity management solution and need to be well designed and implemented.

IBM Tivoli Directory Integrator as identity data feed tool

Tivoli Directory Integrator is designed to synchronize identity data located in directories, databases, collaborative systems, applications used for Human Resources (HR), customer relationship management (CRM), Enterprise Resource Planning (ERP), and other corporate applications. In Identity Manager, a provisioning service type called an *IDI Data Feed* is supported for user data exchange between Directory Integrator and the Identity Manager server, it is based on the DSMLv2 protocol.

Each identity data feed process will be a Tivoli Directory Integrator AssemblyLine. The AssemblyLine can pull data from other sources to complete identity attributes if they are not all in the same system. Figure 19-13 on page 605 shows an example of an identity data feed that pulls data from multiple sources. Note that even data is pulled from other data sources, the HR system is considered the authoritative identity data source.

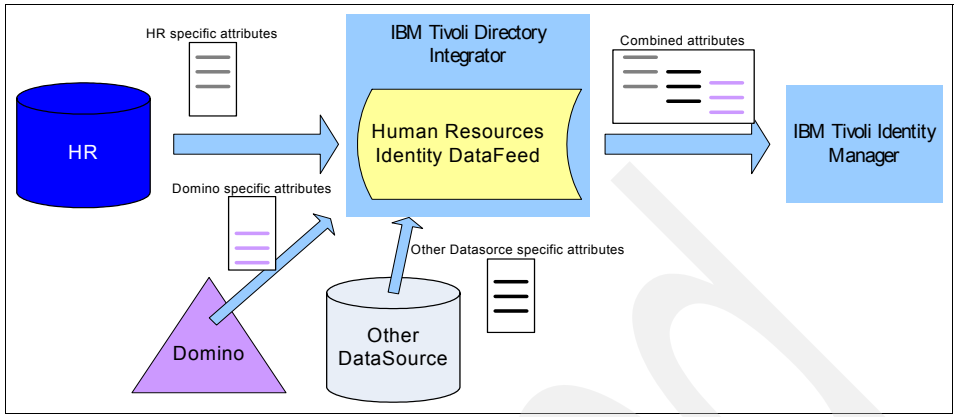


Figure 19-13 Identity data feed example

It is also very common to have multiple authoritative identity data sources. For example, if a company manages identities for employees, contractors and customers, probably this implementation will have three authoritative identity data sources. Figure 19-14 shows that three authoritative identity data sources mean that three Tivoli Directory Integrator AssemblyLines must be developed.

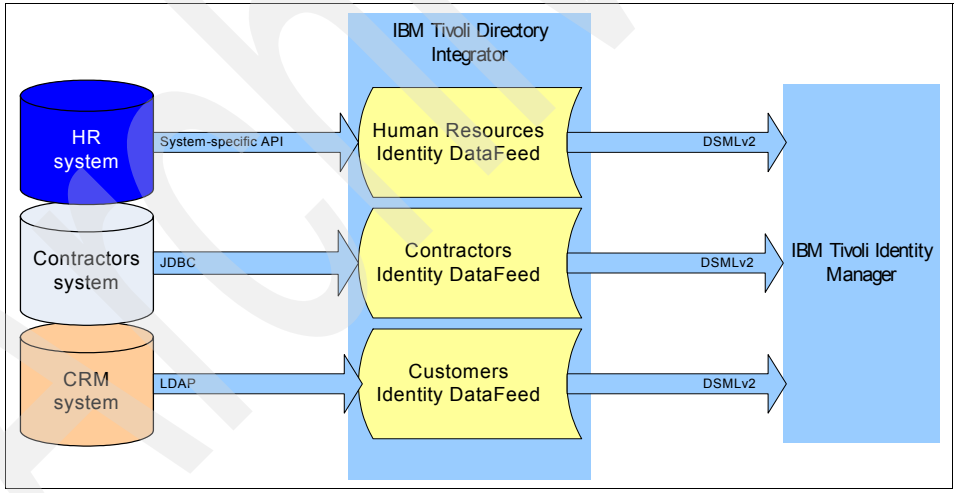


Figure 19-14 Multiple authoritative identity data sources

Please note that it is possible, but uncommon, to have just one AssemblyLine that pulls data from multiple identity data sources. It can be done if data from these identity data sources and the process are very similar.

The identity data feed is not only about feeding identity data into Tivoli Identity Manager. It is also about making decisions based on this data that will be a base for the person entity's lifecycle management. Table 19-14 shows some examples of authoritative identity data sources that triggers lifecycle management events

Table 19-14 Identity data source events and corresponding lifecycle management events

Event type	Lifecycle Management event
Vacation start	Suspend person entity and all its accounts
Vacation ends	Restore person entity and all its accounts
Employee terminated	Delete person entity and all its accounts
Contract expired	Suspend all person entities and its accounts of this contract

Areally Big Investment Corp has three authoritative identity data sources for its main location in United States for employee, contractor and customer identity management. They also have other country specific authoritative identity data sources for employee and contractor identity management.

Tivoli Directory Integrator can be deployed in a central location. But depending on network speed and complexity of the communications, it may also be deployed in remote locations and from there push data into Tivoli Identity Manager. As seen in Figure 18-7 on page 568, Tivoli Directory Integrator and Tivoli Identity Manager communicate using DSMLv2 over HTTPS, so the Tivoli Directory Integrator location for this task is very flexible and firewall friendly. As Tivoli Directory Integrator is simple to deploy and its license is based on what it manages, it is not uncommon to have a dozen Directory Integrator instances deployed based on best performance, security, and ease of administration.

Figure 19-15 on page 607 shows Areally Big Investment Corp identity data feeds. Note that in Brazil they have a Lotus Notes contractors system and for some Lotus Notes performance issue across the WAN they decided to install a Tivoli Directory Integrator server in the Brazil production zone.

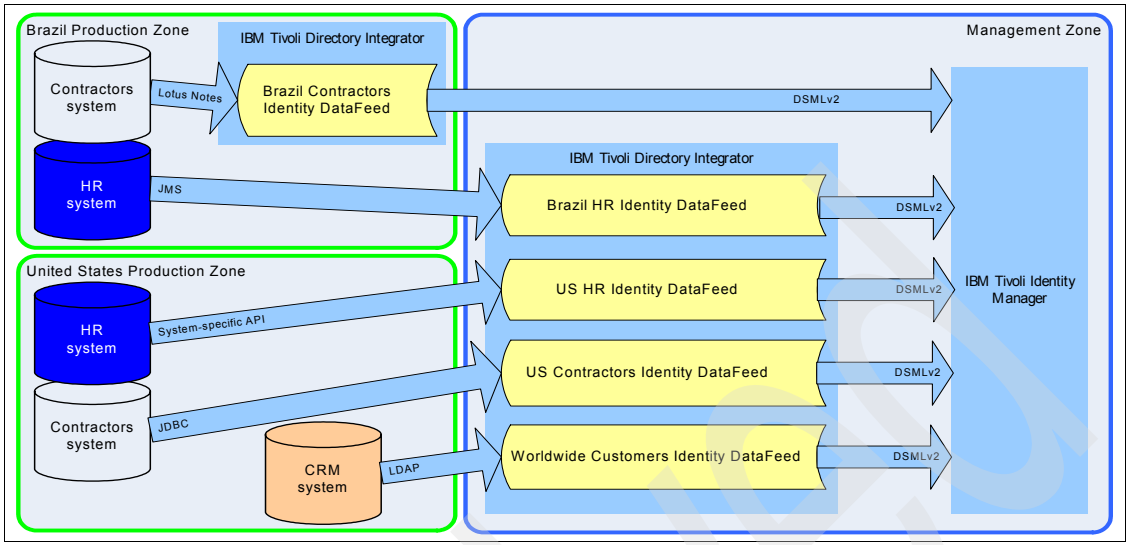


Figure 19-15 Areally Big Investment Corp identity data feed

Tivoli Directory Integrator is a high-value solution to integrate Tivoli Identity Manager with business processes and identity data.

19.6 Integrating with Access Manager

As we discussed earlier in this book, IBM Tivoli Access Manager for e-business provides policy-based *access control enforcement* for enterprise applications, Web applications, and resources. IBM Tivoli Identity Manager provides policy-based *identity management* (managing user IDs and passwords) and *provisioning* (providing or revoking access to applications, resources, or operating systems) within an enterprise. The important point here is that they can and should be combined to utilize the specialized security features of both.

The major consideration when combining these environments is that you will continue to manage access control for applications and resources using Access Manager, but you will use Identity Manager to manage Access Manager users and to manage the provisioning of applications and resources to those users. Identity Manager becomes the central repository for information, as depicted in Figure 19-16 on page 608.

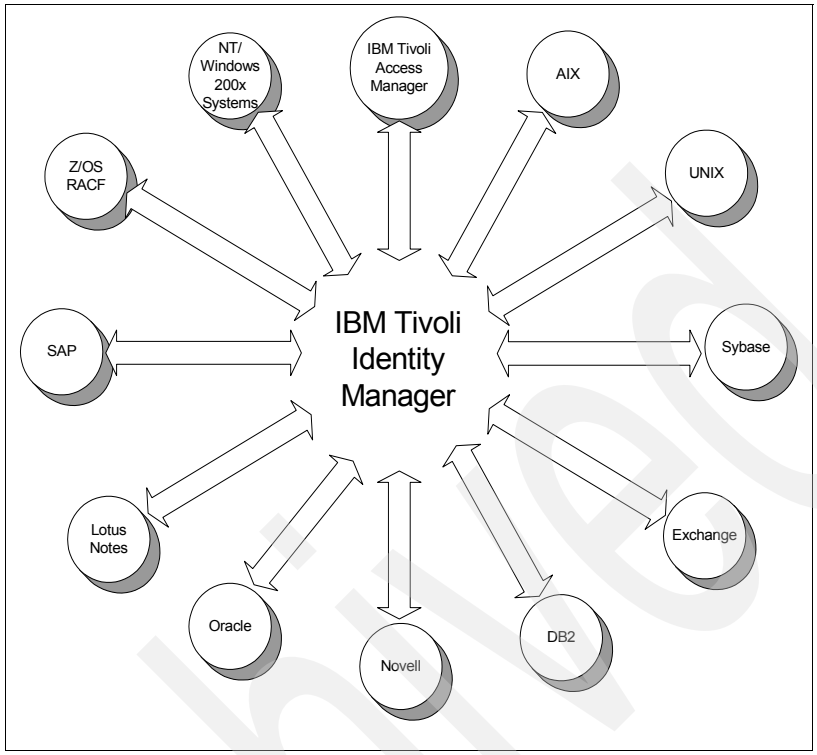


Figure 19-16 Identity Manager as the central repository for user information

To link these products, you must perform some basic integration tasks and some Identity Manager tasks.

Integration possibilities are:

- Tivoli Identity Manager manages Tivoli Access Manager users and groups, exactly the same way it's done with all platforms managed by Tivoli Identity Manager.
- Tivoli Access Manager WebSEAL protects Tivoli Identity Manager Web interface, also doing single sign-on between products.
- Password changes executed with a WebSEAL password change transaction are replicated to Tivoli Identity Manager. This feature is implemented by the Reverse Password Synchronization for Tivoli Access Manager adapter.
- Data synchronization between Tivoli Access Manager user database and Tivoli Identity Manager.

The integration of Identity Manager with Access Manager requires that we consider the placement of all of their combined components. Figure 19-17 on

page 609 shows a sample architecture for integrating Identity Manager and Access Manager. It shows, by zone, the recommended placement of main Access Manager components.

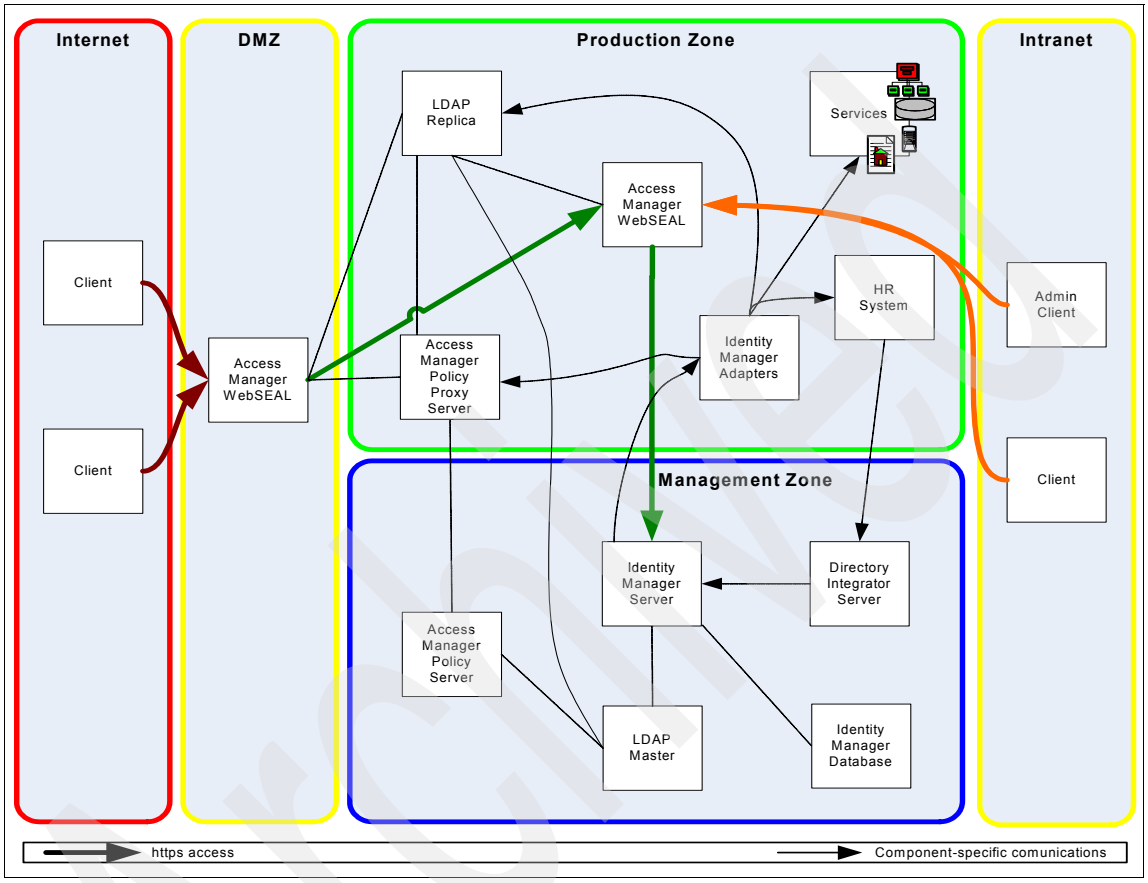


Figure 19-17 Integrated architecture for Access Manager and Identity Manager

To achieve integration after installing Identity Manager in your Access Manager environment, you should use Tivoli Identity Manager and its interface instead of the Access Manager interfaces to manage the Access Manager system users, exactly the same way as for other platforms.

If Identity Manager has to manage more than one Tivoli Access Manager secure domain, you should create a service for each Tivoli Access Manager domain

To manage global sign-on access (that is, GSO resources and GSO resource groups), the Tivoli Access Manager GSO Adapter is needed. This adapter enables you to create services for GSO resources and GSO resource groups.

19.6.1 Specialized integration tasks

Depending on the complexity of your integrated environment or your existing Tivoli Access Manager system, you might need to complete specialized tasks that are related to the integration. Some examples of specialized tasks include:

- Configuring Tivoli Identity Manager for single sign-on with WebSEAL.
- Synchronizing Tivoli Identity Manager user data with Tivoli Access Manager user data.
- Creating a Web interface from which users can self-manage their user IDs and passwords and request access to applications or resources.
- Deploy an LDAP adapter to manage attributes not managed by Tivoli Identity Manager.

19.6.2 Integrated architecture with Identity Manager adapters

When the services are connected and integrated, Identity Manager becomes the focal point for all user management disciplines, and Access Manager is the focal point for access control disciplines. Note that in Figure 19-17 on page 609 the Identity Manager server talks to the adapters in the Access Manager server environment and not to the Access Manager servers themselves. Access Manager is a service managed by Identity Manager. Identity Manager can utilize the information that is stored in the Access Manager server environment. In other words, Identity Manager has the capability to make password and all other changes to the Access Manager environment.

With the deployment of Tivoli Access Manager Password Synchronization Adapter, WebSEAL has the ability to check the password strength rules set in Identity Manager and initiate the password change. Tivoli Identity Manager then propagates the password change to other accounts managed. This ensures that regardless of whether users change their password through Tivoli Identity Manager or Tivoli Access Manager WebSEAL, password synchronization can be maintained.

19.7 Conclusion

Alone, Tivoli Identity Manager is a powerful enterprise application for managing the lifecycle of persons, accounts, and managed services from many different facets. When coupled with Tivoli Access Manager, a powerful identity and access control management foundation becomes available for the enterprise.

Tivoli Directory Integrator helps integrate current technology and business processes into Tivoli Identity Manager and Tivoli Access Manager and it is a key component to integrate diverse enterprise data sources and directories and keep the information properly synchronized.

Identity Manager, Directory Integrator, and Access Manager offer extensive opportunities to solve complex systems management and security needs.

20

Identity Manager Express structure and components

In this chapter we describe the IBM Tivoli Identity Manager Express concepts and components. This includes the provisioning strategies employed by our customers and where they apply to different contexts, the logical and physical components, and the mechanisms to keep the solution secure.

At the heart of any identity management solution is the process of provisioning and de-provisioning of accounts. IBM Tivoli Identity Manager Express introduces a different provisioning paradigm to that used by Tivoli Identity Manager. Tivoli Identity Manager Express provides a request based provisioning solution, as opposed to using policy-based provisioning as per Tivoli Identity Manager. At the heart of the product differences, this is the main shift in focus.

Note: This product is targeted at the small-to-medium business, with a supported limit of 5000 users per deployment.

20.1 Provisioning strategies for identity management

Identity management systems today approach the provisioning process in different ways. In this section, we describe two of these approaches: *policy-based provisioning* and *request-based provisioning*. Each approach has its advantages and challenges. A number of factors determine which approach is best suited for your implementation. We discuss these factors in this section. If your organization is small, determine the best provisioning solution based on your organization's needs and in providing them with quick time to value.

20.1.1 Policy-based provisioning

On one end of the spectrum, there is the concept of policy-based or role-based user provisioning. A popular term behind this principle is *Role-Based Access Control* (*RBAC*).

RBAC is the process of granting access privileges to the users based on the work that they do within an organization. This allows an administrator to assign the users to one or more roles according to the job they do. Each role enables access to specific resources based on a provisioning policy. Accounts or access rights are granted to the role rather than to the user. A user has to be a member of the role to be granted that account or access privilege.

Roles can be defined roughly covering a broad range of users, or they can be finely tuned to cover many types of account and access rights. An example of a rough classification of roles is employees versus contractors.

In the case of role-based provisioning, a significant amount of effort is spent in the initial policy and role design to automate the provisioning processes. However, when the policy design and roles are implemented, a high degree of automation can occur in the management of the identity lifecycle.

- Benefits
  - High degree of automation
  - Quickly add and revoke privileges based on role changes
- Challenges
  - Role engineering can be complex
  - May not scale if too many user requirements are unique
  - Too many roles have to be defined

Role-Based Access Control is discussed in 17.6, "Access control models" on page 527.

20.1.2 Requests-based provisioning

On the other end of the spectrum, there is request-based user provisioning. This method uses centralized management but decentralized administration. That is, the users are responsible for requesting the account access they want to receive. More operational labor is required because no automation of the provisioning processes exists with request-based provisioning. Most systems implement a workflow component to provide approvals for access rights.

- Benefits
  - Easier, less expensive, and faster to implement
  - Managers, application owners, and administrators control access rights through approval process
- Challenges
  - Users might not always know what they require
  - May not scale in large environments based on the manual effort required

20.1.3 Combining policy-based and request-based provisioning

A composite approach is a hybrid of the two approaches. An example is where temporary employees can be provisioned a set of services based on their roles, and permanent employees request what services they want to receive. Some elements of the roles are necessary as you move from manual to automated provisioning.

An organization might want to start realizing the benefits of an identity management system by implementing a request-based provisioning system first, and then move to a policy-based system in a phased approach as the processes and requirements become more well defined.

20.1.4 Features of IBM Tivoli Identity Manager Express

Identity Manager Express provides a request-based provisioning approach to grant, modify, and remove access to resources throughout a business or business unit, and to establish an effective audit trail using automated reports. Users, or their managers, can request access to new accounts. Additionally, managers or other administrators are alerted to unused accounts and given the option to delete the accounts through a recertification process. This recertification process ensures that over time users do not accumulate more access rights than they require.

Identity Manager Express is designed for small-to-medium sized businesses and decentralized departmental usage in large companies with 100 to 5000 users. In

the next section, we describe the logical components of an identity management system and those that are specific to Identity Manager Express.

20.2 Management and user terminology

Tivoli Identity Manager Express shares common definitions of entities with the Tivoli Identity Manager product. The reader is encouraged to read the following sections (rather than reproducing the content exactly here) provided within the Tivoli Identity Manager chapter in order to grasp the concepts described later:

- People, person, accounts and user: For a complete description of person, people, accounts and users, please refer to 18.1.1, "Users, accounts, and attributes" on page 548. Keep in mind that Tivoli Identity Manager Express provisioning is request-based.
- Identity Feed: For a complete description of identity feed, please refer to 18.1.2, "Identity feed" on page 550.
- Passwords: for a complete description of passwords, please refer to section 18.1.3, "Passwords" on page 550.

Note: Some managed resource adapters can capture a password as it is being changed directly on the managed resource and then pass it on to Identity Manager Express for password synchronization. This requires the installation of a plug-in on the managed resource. This process is know as *reverse password synchronization*. Only one instance of reverse password synchronization can be enabled for a deployment of Identity Manager Express.

The following Identity Manager Express adapters enable reverse password synchronization:

- Tivoli Access Manager
- Microsoft Windows Server® Active Directory
- IBM AIX
- IBM AS/400®

Changing a password in the *master password store* changes all the passwords on accounts on other resources that Identity Manager Express manages for that same user. The synchronization occurs irrespective of whether the Identity Manager Express password synchronization is off or on.

- Service: For a full description of a service, please refer to section 18.1.5, "Managed systems and applications" on page 551.

The following sections outline those areas where Tivoli Identity Manager Express contains subtle differences to its Tivoli Identity Manager product partner.

20.2.1 Setting policies in Identity Manager Express

Identity Manager Express provides for the definition of an *identity policy* and a *password policy*. These policies can be defined at a system level (global) or at a service-specific level.

Password policy

A password policy defines the rules that determine whether a new password is acceptable. It sets the rules that passwords for a service must meet, such as length and type of characters allowed. Additionally, the password policy might specify that an entry is disallowed if the term is in a dictionary of unwanted terms. To select this choice in the user interface, you must first load a dictionary.ldif file into the Identity Manager Express server.

A password strength rule is a rule to which a password must conform. For example, password strength rules might specify that the minimum number of characters of a password must be five and the maximum number of characters must be ten. You can specify the following rules for passwords:

- Minimum and maximum length
- Character restrictions
- Frequency of password reuse
- Disallowed user names or user IDs

Identity policy

An identity policy defines how a user's ID is created. Identity Manager Express automatically generates account user IDs from the identity policy. Identity policies can be set as a global policy for all accounts or as a service-specific policy. For example, if all the user IDs for all accounts must be composed of the user's first initial and last name, a global identity policy must be created for the organization. If all user IDs for a specific service must contain a certain number, a service-specific identity policy must be created for the service.

20.2.2 User categories

Identity Manager Express provides different categories of users, which are used to define the default permissions and operations, and the views of the Identity Manager Express application that the user can access. For each category of user, Identity Manager Express defines default *access control items (ACI)* and default *views* that the users can access.

Identity Manager Express provides the following categories of users:

- User
- Manager
- Help desk assistant
- Service owner
- System administrator

Each category, except for user, has a corresponding group defined. All Identity Manager Express users are automatically part of the user category and are granted the base level of permissions and access to the base set of views.

For each category of user, except the system administrator category, you can customize the views that are available to the users and create customized groups based on that category of user. The users in the default system administrator group always have access to all the views and can perform all operations in Identity Manager Express. You cannot modify a category.

Categories have relationships with groups, access control items, and workflows, which are defined in Table 20-1.

Table 20-1 User category relationships

Category	Description
Manager	Members of the manager group are users who manage the accounts, identity profiles, and passwords of their direct subordinates, unless the person form is customized to exclude some of the attributes for which the manager has permission to read or write. Managers can manage and delegate activities on their to-do lists.
Service owner	Members of the service owner group manage a service, including the user accounts and requests for that service. Additionally, on services they own, service owners can view others' requests, such as authorizing an account, unless the person form is customized to exclude some of the attributes for which the service owner has permission to read or write. Service owners can manage and delegate activities on their to-do lists.
Help desk	Members of the help desk assistant group can change or reset others' passwords, profiles, and accounts, unless the person form is customized to exclude some of the attributes for which the help desk assistant has permission to read or write. Additionally, help desk assistants can restore accounts, and also view others' requests, and both manage and delegate to-do lists.
User	Users have basic privileges on their own identity and account information. They can request accounts and change passwords.

Category	Description
System administrator	The system administrator performs both security and system administration tasks. An Identity Manager Express administrator has access to the complete portfolio of functions and tasks.

20.2.3 Access control

Access control topics include views, groups, and access control items (ACIs).

Views

A view is a set of tasks that users can perform. The view defines what tasks are available and visible when they use Identity Manager Express.

Groups

A group is a collection of users. Users can belong to one or more groups. Groups are used to control user access to functions and data in Identity Manager Express. Users can belong to default groups that Identity Manager Express provides or you can also create additional, customized groups.

Groups grant specific access to functions and resources within Identity Manager Express. For example, one group might have members who work directly with data defined in a business application. Another group might have members who provide a subset of the Help Desk group functions.

A user with no group membership sees a user interface that has tasks only for the user category. A user with membership in a group can see an expanded user interface that has tasks for the user and additional group tasks.

Using the groups

An ACI is defined to specify a set of operations and permissions, and then identify which groups are governed by the ACI. A workflow is defined to specify an approval cycle for account requests, and then identify one or more groups as participants.

Access control item

An ACI is data that identifies the permissions that users have for a given type of resource. You create an ACI that allows you to specify a set of operations and permissions, and then identify which groups use the ACI.

An ACI defines the following items:

- The entity types to which the ACI applies
- Operations that users can perform on entity types

- Attributes of the entity types that users can read or write
- The set of users that is governed by the ACI

For example, you can create an ACI that prevents the help desk group from creating or deleting users.

20.2.4 Workflow

A workflow defines the sequence of activities that represent a business process. Workflows are used to provision accounts according to your specific business practices. They generate to-do items that appear in the users activity list. Identity Manager Express supports two types of workflows: *account request workflows* and *recertification workflows* or *policies*.

Account request workflows

An account request workflow defines the activities for managing requests for accounts. The workflow can consist of activities that route a request for approval, provide e-mail notifications, or cause requests for information to occur.

Recertification policies

Identity Manager Express recertification simplifies and automates the process of periodically revalidating user accounts and ensuring that users still have the appropriate privileges. The recertification process automates the validating process that each user account is still required for a valid business purpose. The process sends recertification notification and approval events to the participants who are specified in the policy.

The following actions can be taken on an account recertification activity. After a recertification request occurs, an account can be as follows:

Active If the recipient takes no action and the approval times out, the account remains active.

Suspended A recipient declines recertification. The workflow suspends the account and issues suspension notifications.

Deleted A recipient declines recertification. The workflow deletes the account and issues deletion notifications.

20.3 Physical component architecture

The Identity Manager Express solution includes the Identity Manager Express server, its required middleware components, and resource adapters. Deploying Identity Manager Express requires a single-server configuration that includes all the middleware components described in this section.

Figure 20-1 presents all the components of a typical Identity Manager Express implementation.

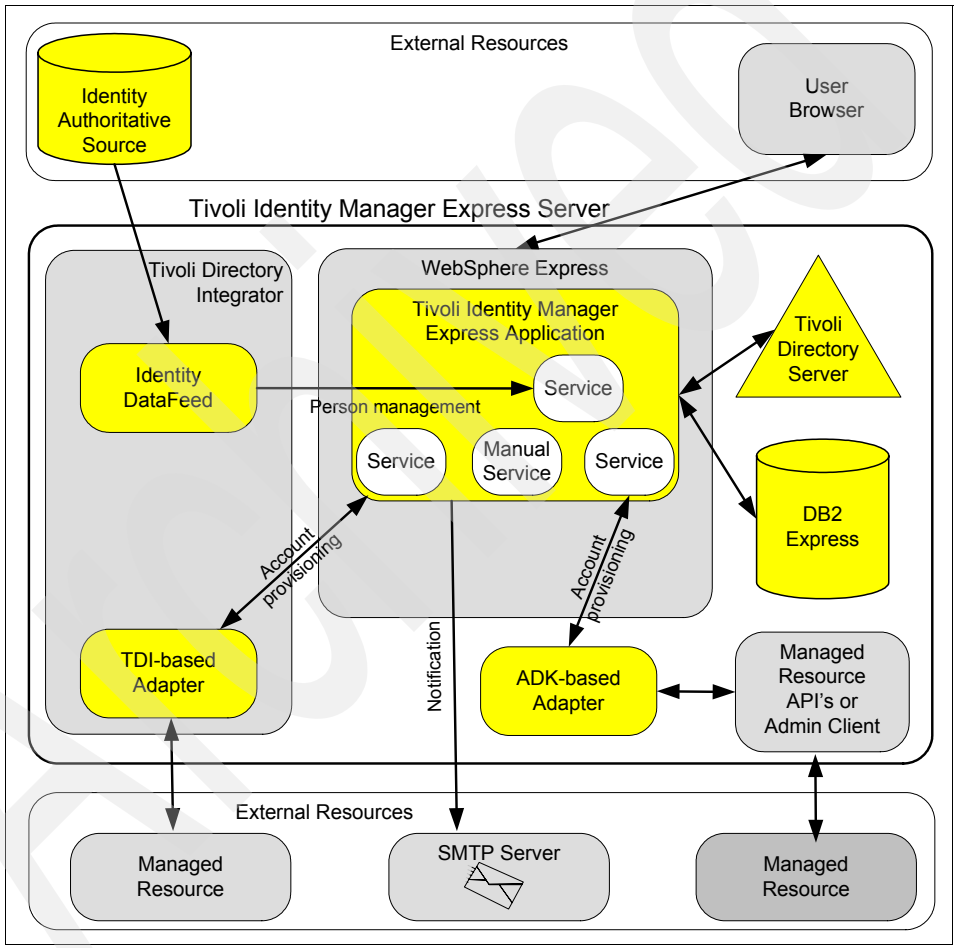


Figure 20-1 Identity Manager Express architecture overview

Identity Manager Express components

This section examines the single components in more detail.

IBM Tivoli Identity Manager Express Server

The IBM Tivoli Identity Manager Express 4.6 Server and its adapters enable you to provision accounts to a set of computing resources, which can be operating systems, data stores, or other applications. The Identity Manager Express application is a Java 2 Platform, Enterprise Edition (J2EE) application that runs on the IBM WebSphere Application Server Express.

IBM WebSphere Application Server Express

WebSphere Application Server is the primary component of the WebSphere environment. It runs a Java Virtual Machine (JVM™) providing the runtime environment for the enterprise application code, communication security, logging, messaging, and Web services.

IBM DB2 Express database

Identity Manager Express stores transactional and historical data in the IBM DB2 Universal Database Express Edition Server, a relational database that maintains the current and historical states of data. Every transaction done is placed here and is used for the transactional purpose of current processes and historic data for auditing purposes.

IBM Tivoli Directory Server

Identity Manager Express stores the current state of the managed identities in IBM Tivoli Directory Server, an LDAP directory. This includes user account and Identity Manager Express application configuration data such as policies and its own access control mechanism.

IBM Tivoli Directory Server is discussed in 3.3, “IBM Tivoli Directory Server” on page 72.

IBM Tivoli Directory Integrator

IBM Tivoli Directory Integrator can synchronize data in directories, databases, and other repositories. This eliminates the need for a central data store and provides flexible connection of data from repositories throughout an enterprise.

IBM Tivoli Directory Integrator is installed to run as a service on the Identity Manager Express server to provide adapter communications. Adapters that are created using Directory Integrator are implemented as Directory Integrator AssemblyLines. Each of these lines is a single path of data transfer and transformation. The following section on adapters contains more information about Directory Integrator based adapters, called *Tivoli Directory Integrator adapters*.

Directory Integrator is also used for integration of one or multiple identity data sources. Most implementations have at least an integration with the human resources system.

IBM Tivoli Directory Integrator is discussed in 3.5, “IBM Tivoli Directory Integrator” on page 96.

Adapters

Several Directory Integrator based agentless adapters are automatically installed when you install Identity Manager Express. You can install additional agentless or agent-based adapters that are either Directory Integrator based or ADK-based.

For more information about adapters, please reference 18.3.6, “Resource connectivity” on page 566.

20.4 Identity Manager Express security

The Identity Manager Express environment can be secured at every component level. Although it is a single server identity management solution, it provides the following benefits:

- Manages several distinct managed resources
  - Uses encryption between Tivoli Identity Manager adapters and managed resources where necessary.
  - If remote adapters are in use, you can configure adapter access control.
- Because the Identity Manager Express environment is potentially accessed by different types of users from different places, sometimes from insecure networks, configure the following two mechanisms:
  - Encryption
  - Another layer of Web access security
- The Identity Manager Express environment requires *near exclusive* access to the managed resources

  To use most of its value as a single point of management and auditing solution, enforce its use as the only identity management solution.

In the following sections, we discuss each one of these points, internal components security, and Identity Manager Express server access security.

Identity Manager Express server access security

All the components and installed adapters are typically located in a single server configuration. When there are many components talking to each other through TCP/IP, enabling encryption between them seems logical. However, because all communications between components occur on the same physical machine and are not transmitted over any kind of network, you can have good security with a simple setup.

It is possible to have a fully working Identity Manager Express implementation with only the Web server port open for incoming connections. The communications between Identity Manager Express and managed resources always originate from the Identity Manager Express server. Therefore, we can accomplish Identity Manager Express server security by following these simple rules:

- Secure Identity Manager Express server physical access

  Prevent easy access by non-authorized personnel.

- Block all incoming connections to the Identity Manager Express server, with the following exceptions:
  - Web server plug-in or reverse proxy connections
  - Push components such as password synchronization plug-ins or Directory Services Markup Language v2.0 (DSMLv2) data feed

This allows users to gain access to the Identity Manager Express application, but no other type of access is allowed, specifically to the LDAP and DB2 components, which are the components safeguarding all data.

Security for managed resources

Because we use TCP/IP communications between Identity Manager Express and its adapters, and because all managed resources are remote, we have to enable security between all types of adapters, Directory Integrator-based or ADK-based.

Figure 20-2 shows a typical communication pattern between the Identity Manager Express application and the managed resources. For each adapter implemented, apply the managed resource specific security configurations.

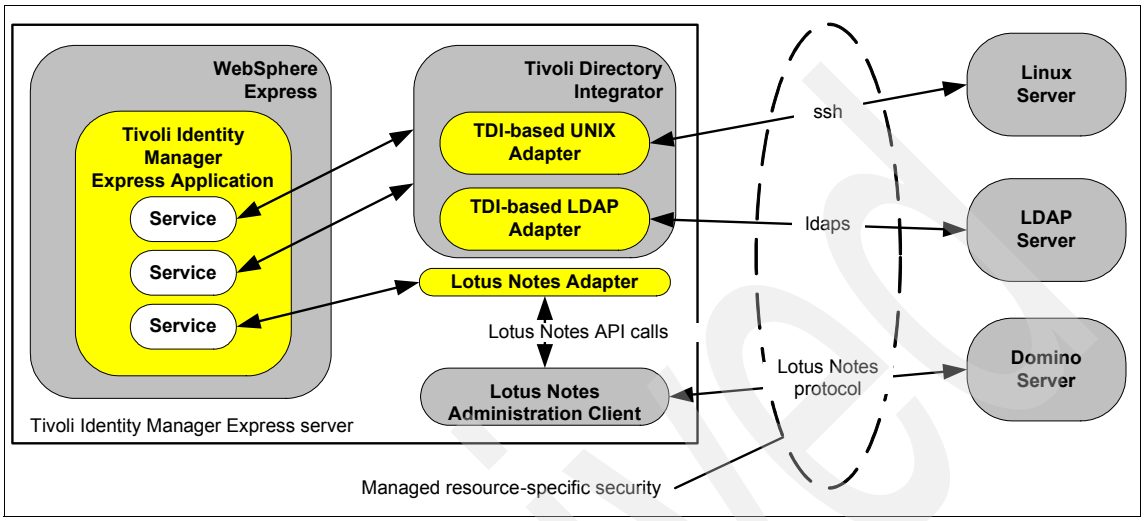


Figure 20-2 Typical communication between Identity Manager Express and its resources

In this case, we have Linux, LDAP, and Lotus Notes or Domino adapters deployed. Apply the following managed resource specific configurations:

- Linux adapter

  The Linux adapter uses the SSH protocol, which is already a secure protocol.

- LDAP adapter

  Configure the LDAP server and LDAP adapter to use LDAP with Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol, which is called *LDAPS*. It is also best practice to use an exclusive LDAP account for the LDAP adapter.

- Lotus Notes adapter

  The Lotus Notes adapter issues API calls to the locally installed Notes Administration Client. To enable security, the Lotus Notes Administration Client must have encryption enabled so that all communications from the adapter to the Domino server are secured.

The same applies to other adapters. However, there are some cases where encryption is not available, or the nature of the managed resource requires some local code running.

Figure 20-3 shows two common examples, where accounts are stored in file-based databases and files.

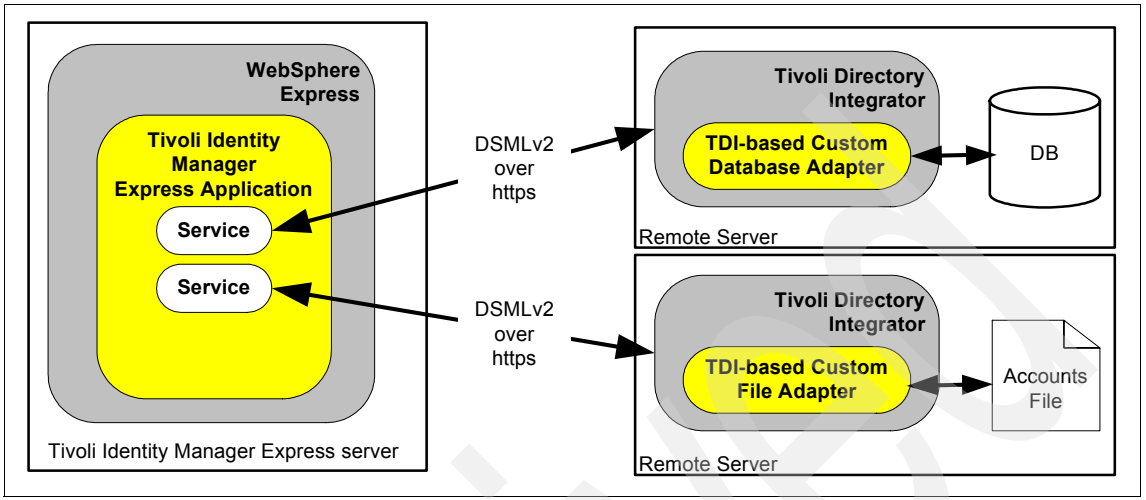


Figure 20-3 Remote communications with Tivoli Directory Integrator-based adapters

In these cases, there are no possible remote communications or the available ones are insecure. Therefore, the custom developed Tivoli Directory Integrator-based adapter has to be deployed together with the managed resource. Identity Manager Express uses DSMLv2 over Hypertext Transfer Protocol-Secure (HTTPS) to communicate with the remote Tivoli Directory Integrator-based custom adapter.

To have the highest possible security in this scenario, ensure the following prerequisites:

- Ensure managed resource server access. The custom adapter is an Extensible Markup Language (XML) file and it is a best practice to protect it.
- Enable password authentication at the custom adapter so that it requires a corresponding user name and password at the Identity Manager Express service configuration.
- Set firewall rules so that only the Identity Manager Express server can connect to the DSMLv2 Hypertext Transfer Protocol (HTTP) port configured for the custom adapter.
- Enable SSL security at the custom adapter.

The same prerequisites are valid for remote ADK-based adapters.

Identity Manager Express Web security

WebSphere Application Server Express Edition uses its own Web server. However, add another HTTPS security layer so that browsers do not have direct access to the Identity Manager Express Web server port. There are two options:

- Deploy a remote Web server and configure it with the WebSphere Application Server plug-in.
- Deploy a reverse Web-proxy such as Tivoli Access Manager for e-business WebSEAL.

Figure 20-4 shows a typical scenario using an IBM HTTP Server as a middle tier between browsers and Tivoli Identity Manager Express.

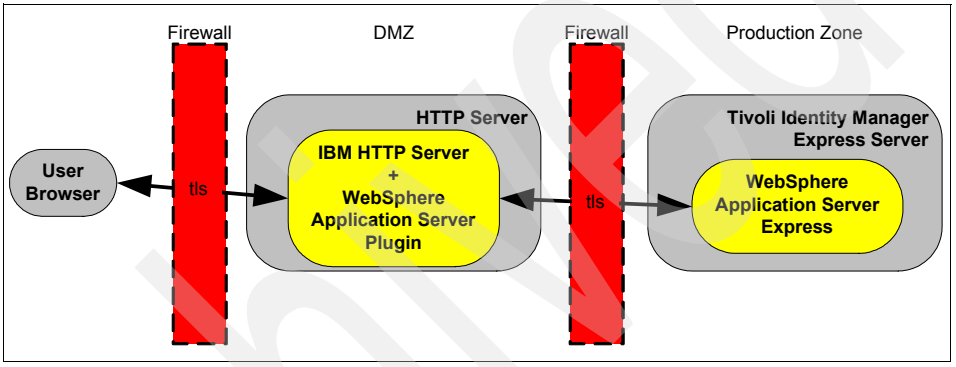


Figure 20-4 Three-tier Web access using IBM HTTP Server

Identity Manager Express framework security

"Identity Manager Express server access security" on page 624 shows that it is possible to have good middleware security locking the Identity Manager Express server machine itself. However, it is possible to implement this by enabling security at each one of the middleware components.

WebSphere Application Server security

The WebSphere Application Server installation program selects OFF as the default value for WebSphere Application Server global security. However, your environment might require that you provide WebSphere Application Server global security. When enabled, WebSphere Application Server global security ensures that authenticated users have the necessary permissions to access Tivoli Identity Manager Express Enterprise JavaBeans™ (EJB) components.

Configuring this security component involves configuring an authentication mechanism, a user registry, and optionally, Java 2 security. There are two types of security to consider:

- WebSphere Application Server global security

  Global security is primarily concerned with application security and enforces authentication and role-based authorization. When global security is enabled, you cannot log on to the WebSphere Application Server administration console without a user ID and password.

  Enabling global security introduces two important IDs to the WebSphere Application Server environment:

  – The server user ID

    Basically this is a user in a user registry such as an LDAP or local operating system user. The user is a member of the chosen user registry, but also has special privileges in WebSphere Application Server. The privileges for this ID and the privileges associated with the administrative role ID are the same. The server user ID can access all protected administrative methods.

    On Windows systems, the ID must not be the same name as the machine name of your system, because the registry sometimes returns machine-specific information when querying a user of the same name. In LDAP user registries, verify that the server user ID is a member of the registry and not just the LDAP administrative role ID. The entry must be searchable.

  – The process ID

    The WebSphere Application Server processes are run by the process ID rather than the server user ID. The process ID is determined by the way the process starts. For example, if you use a command line to start processes, the user ID that is logged into the system is the process ID. If running as a service, the user ID that is logged into the system is the user ID running the service.

    If you choose the local operating system registry, the process ID requires special privileges to call the operating system APIs. Specifically, the process ID must have the *Act as Part of Operating System* and *administrator* privileges on Windows systems or *root* privileges on a UNIX system.

- WebSphere Application Server Java 2 security

  Java 2 security can optionally be turned on or off when global security is enabled. It addresses the use of system resources such as writing to the file system, listening on a socket, and calls to APIs. Java 2 security is configured in a was.policy file.

Enabling Java 2 security for the Tivoli Identity Manager Express application causes Java 2 security to be enforced on all applications that are running on the WebSphere Application Server. If you enable Java 2 security for the Tivoli Identity Manager Express application, you must also appropriately configure all other applications running on the WebSphere Application Server to support Java 2 security.

The Java 2 security policy that Tivoli Identity Manager Express provides grants Tivoli Identity Manager Express all permissions on the system. It does not bring any security benefit for Identity Manager Express deployments mainly because Identity Manager Express is always a single and dedicated WebSphere deployment scenario.

Enabling Java 2 security can also cause some reduction in performance of the WebSphere Application Server between 10% to 20%. If you have to configure Java 2 security, refer to the *IBM Tivoli Identity Manager Express Installation Guide V4.6*, SC32-2262.

If the chosen scenario follows the recommendations given in “Identity Manager Express Web security” on page 627, and there are no open communications to the Identity Manager Express server other than Identity Manager Express’ own components, you can choose to not enable security at all for WebSphere Application Server.

WebSphere Application Server Web server security

Block all incoming traffic to the Web server except from the WebSphere Application Server plug-in deployed together with the HTTP server.

Directory Server security

You can easily make the Tivoli Directory Server that comes with Identity Manager Express secure, if it is not used by any other applications, by performing the following steps:

1. Block all incoming connections to LDAP ports. The default ports are as follows:
   - 389 for plain and TLS LDAP connections
   - 636 for LDAP over SSL/TLS connections
   - 3538 for the Tivoli Directory Server administration daemon
2. Disallow anonymous binds.

However, if this LDAP server is also used by external applications, perform the following steps to make it secure:

1. Block all incoming connections to the Tivoli Directory Server administration daemon on port 3538.
2. Enable and enforce SSL connection to it.

3. If not necessary, disallow anonymous connections.
4. Create Tivoli Directory Server access control lists (ACLs) to prevent someone from reading Identity Manager Express data.

DB2 security

The DB2 database that comes with Identity Manager Express must not be used by any other applications. You can easily make it secure by blocking all incoming connections to DB2 listening ports. The default ports are as follows:

- 3700 for the DB2 instance used by Tivoli Directory Server
- 50000 for the DB2 instance user by Tivoli Identity Manager Express

Administrative password security

Identity Manager Express has full administrator access to all managed resources. Getting access to Identity Manager Express as an administrator grants access to any type of account creation on any platform. Therefore, it is important to choose and maintain good and secure passwords for Identity Manager Express administrators.

Managed resource security

If deployed and in production, Identity Manager Express is considered to be the only system to manage accounts. To take advantage of its capabilities and security features, such as central auditing, disable all access at managed platforms for account operations.

For example, if you have help desk users with account operator rights when implementing Identity Manager Express, you can provide them access to the Identity Manager Express help desk group and revoke their special privileges on individual managed resources. This improves performance and ensures that nobody manages accounts directly on the managed resources.

Adapter security

The Tivoli Identity Manager Express Server uses either SSL or SSH communication to communicate securely with supported adapters. The following measures protect adapters from misuse:

- Ensure that only the Identity Manager Express host can connect to the adapter listening TCP port.
- Configure each remote adapter to use SSL. Refer to instructions about each adapter to enable it.
- On ADK adapters, choose a good password for the following:
  - Adapter configuration tool access (agentCfg)

– Directory Access Markup Language (DAML) protocol users and passwords

20.5 Conclusion

This concludes the discussion on the Identity Manager Express structure and components. You now understand the request-based user lifecycle management approach of the solution and how to provision accounts and people to managed resources. You learned about the logical and physical component architecture of the Identity Manager Express environment and how to secure this infrastructure within your own deployment.

If you want more information about this solution with a scenario discussion and complete deployment solution check out the IBM Redbooks publication *Deployment Guide Series: IBM Tivoli Identity Manager Express 4.6*, SG24-7233.

21

Synchronizing the enterprise

Synchronizing security-related information within an enterprise can be a challenge. IBM Tivoli Identity Manager, Directory Integrator, and Directory Server combine for a viable set of tools for integrating with almost every data repository available in the market today.

In this chapter we map some of the important security architecture attributes to the available IBM Tivoli solutions in the Identity and Credential Management field and provide a variety of customer scenarios that all require some security-related data synchronization.

21.1 Identity data management service context

We identified common security architecture subsystems to provide security functionality and services (as described in 2.1, “Common security architecture subsystems” on page 20). We established five functional categories (see Figure 21-1) that form interrelated subsystems. IBM Tivoli Directory Integrator and IBM Tivoli Identity Manager are products that satisfy component criteria in the Identity and Credential subsystem.

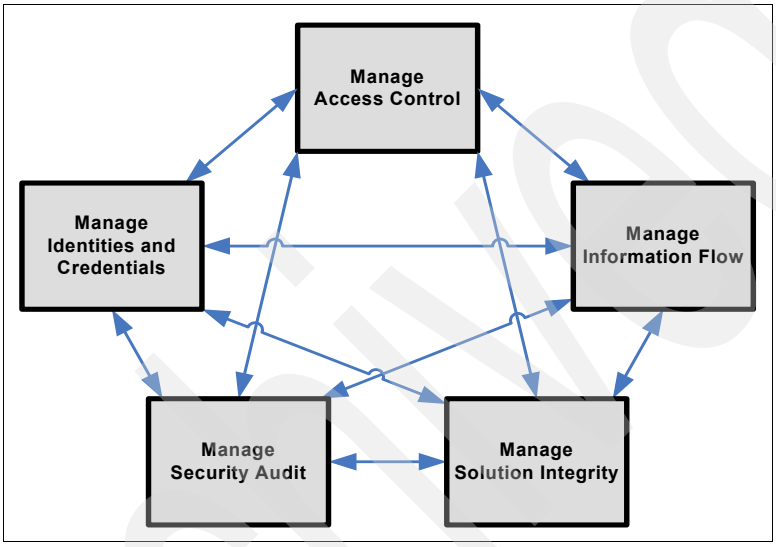


Figure 21-1 Security architecture subsystems

The purpose of a credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms, the processes, and the security subsystems within a computing solution. In some applications, credential systems may be required to adhere to legal criteria for creation and maintenance of trusted identity used within legally binding transactions.

A credential subsystem may rely on other subsystems in order to manage the distribution, integrity, and accuracy of credentials. A credential subsystem has, potentially, a more direct link to operational business activities than the other security subsystems, owing to the fact that enrollment and user support are integral parts of the control processes it contains.

21.2 Identity data repositories

Authoritative identity information is the cornerstone of a secure and efficient enterprise infrastructure and is extremely dependent on a high-performing, highly available security data infrastructure. In this book, we described several applications that rely on directory data for operations (Access Manager, Identity Manager and Federated Identity Manager) in addition to any application within a business system that requires authentication and authorization services. While these application directories are authoritative sources for identity data and resource entitlements in their own domains, other directory sources exist in the enterprise that may be the original source of the information contained within. In fact, research studies found that a typical Fortune 500 customer can have as many as 150 directories.

21.3 Managing identities and credentials

The purpose of the Identity and Credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms and security subsystems within a computing solution. Managing the identity data lifecycle and keeping the data in various repositories synchronized are important components of the Identity and Credential subsystem. Managing identities and credentials, as well as access, must be done in an integrated way in order to protect the integrity of the enterprise. IBM Tivoli Directory Integrator and Identity Manager both provide services to manage identities and data. Depending on the organization's business requirements, security policies, and operational needs, one product may be a better fit than the other or both products may be required.

21.4 Business value

When defining the architecture, breaking down the requirements into the different views, the technologies required for the solution become clearer.

For managing identity data, an integration solution can reduce the administrative burden of managing multiple directories and increase the accuracy of the data. For user provisioning, workflow, and policy enforcement, a more feature-rich application may be required to incorporate business process flows such as approvals. While Tivoli Identity Manager and Tivoli Directory Integrator could be used in similar scenarios, they are distinct and yet complementary products.

Table 21-1 lists some of the components, processes, and properties of the Identity and Credential subsystem. The features of Tivoli Identity Manager and Tivoli Directory integrator are mapped to the groupings.

Table 21-1 Identity and credential mappings

Process/function	Tivoli Identity Manager	Tivoli Directory Integrator
User enrollment	X	X
Manage identities and secrets	X	X
Credential creation	X	X
Credential lifecycle management	X	X
Specification of secrets	X	
Verification of secrets	X	
Credential validation	X	X
Identification/authorization	X	X
Access control		
Information flow control	X	X
Data confidentiality	X	X
Data integrity	X	X
Guaranteed delivery		
Import/export between domains		X
Event awareness	X	X
Event data capture	X	X
Alarms and alerts	X	X
Prioritization of service		
Automated policy enforcement/workflow	X	
Policy enforcement	X	X
Policy administration	X	

Process/function	Tivoli Identity Manager	Tivoli Directory Integrator
Systems management		X
Anomaly handling		X
After-the-fact analysis and reporting	X	

Both solutions can provide significant business value. This can be in the form of cost savings from increased productivity and reduced administration requirements. In addition, data duplication and errors can be reduced through synchronization and policy enforcement, thus providing more data integrity.

The environment's degree of security increases as user IDs are removed from systems and applications when no longer needed due to employee turnover. And as a person's roles are changed within an organization, it is quickly reflected in their access rights to systems and applications. Policy enforcement ensures that corporate security policies are followed.

21.5 Identity data management scenarios

The following section describes some scenarios where the business requirement to provide identity data management was solved through the use of IBM Tivoli Directory Integrator, IBM Tivoli Identity Manager, or both products. The selection of which product to use should come from the analysis of the business requirements, processes or need for processes, functional requirements of the environment, and operational and support requirements for identity data management.

Policies provide the guidance, rules, and procedures for implementing a secure environment. Operational view deals with how the organization does things. Functional requirements are implemented through product features.

21.5.1 Providing metadirectory services

More than a specific product, *metadirectory* is a broad term that covers many technical and business scenarios that have in common a need to combine, reconcile, or synchronize information from multiple identity sources. In one case, a metadirectory solution can consist of a new directory containing information from multiple sources, where the sources are still in control of maintaining the information itself. In another scenario, a metadirectory solution keeps a number

of existing directories synchronized with each other as business requirements keep them separate.

Organizations building e-business applications are becoming increasingly directory-enabled or directory-centric. Identity data can be stored in and used by many applications, all seeking an authoritative data source. But research has shown that large corporations can have more than 100 disparate repositories for user data.

Directory Integrator's strength is providing for the distributed management of information while maintaining data integrity. Directory Integrator can be used as a tool to provide a metadirectory solution that is based on building a new directory or by keeping existing directories synchronized.

The idea is to bring data together from multiple sources and to join the pieces together into a new coherent repository, often called the enterprise directory. Ongoing changes in the source systems are continuously propagated into the enterprise directory so that it always represents a correct and consistent view across the company. From this new source, authoritative data is distributed to all other systems that cannot directly access the enterprise directory.

In the following scenario, a company has a number of disparate systems that contain user IDs, passwords, and other user data such as department information, addresses, and telephone numbers. Keeping all of the systems up to date and synchronized was nearly impossible. The company wanted to develop a central system that housed all of this data and was accessible to employees (for contact information updates and changes) and to administrators (for password resets and additions or deletions). The company also wanted to simplify access to its multiple applications while maintaining high security standards. Password synchronization was desired to reduce help desk calls for forgotten passwords and better security. The solution was to implement a metadirectory that contained information synchronized from each of the separate repositories. This data store is commonly referred to as the corporate directory.

The corporate directory, shown in Figure 21-2 on page 639 as IBM Directory Server, is loaded with data from the Human Resources database. The initial data load imports the entire user population. The data is also merged with existing information in Active Directory and Lotus Domino. This provides data cleanup for the Active Directory and Domino systems.

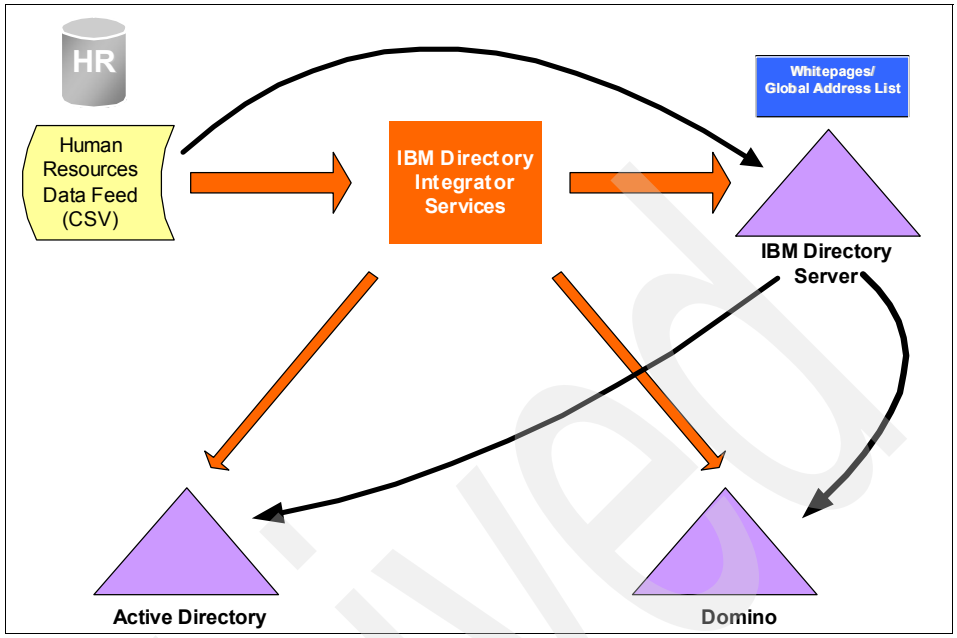


Figure 21-2 Metadirectory services with Directory Integrator

For updates, a nightly extract of the changes is built and presented as a CSV file. When a new user is added to the HR system, the user is created in the enterprise directory as well as Active Directory and Domino. Updates also take place in all three repositories.

Some business logic can be added to take into account terminated employees. For example, if in the HR extract their status is set to *Inactive*, the Active Directory logon will be disabled.

Finally, password synchronization is implemented, and Active Directory will drive all other password changes as being the authoritative password store. When a user changes his Active Directory password (via normal Windows mechanisms) the *Directory Integrator password plug-in for Active Directory* captures the password and updates it in LDAP and Domino.

This is a very common scenario for using Directory Integrator for metadirectory services.

21.5.2 Accelerating Identity Manager deployments

Tivoli Identity Manager can bring significant benefits to an organization by providing centralized identity management.

Identity Manager is used to manage the lifecycle of the person as it relates to the access to systems and applications. Businesses are dynamic and over the course of a person's employment there will be many changes such as promotions, transfers, changing job duties, and perhaps termination due to downsizing, retirement, or resignation.

However, to begin this process, an initial data load must be performed to populate Identity Manager with the Person information of the identities to be managed. This usually involves a large number of entries, so manually entering the data is not an option, and an automated process must be developed. A DSML file could be used to bulk-load the data, but what if the information is coming from more than one source? A custom program could be written, but the goal is to get Identity Manager deployed quickly so the company can begin reaping the benefits that Identity Manager provides in terms of security, efficiency, and productivity.

Directory Integrator can provide the initial data load for Identity Manager identities and maintains this data by synchronizing with an authoritative data source.

In this scenario, a company uses the Human Resources database as the authoritative source for identity information for employees and contractors. Employment status (active, inactive), job roles, location, and supervisor are some of the important attributes about a person that are important to determining the access they need to systems and applications. These attributes can drive role assignments and provisioning policies, so it is imperative that these attributes are updated in a timely manner. A good example is that when a contractor's term expires, access to all accounts should be suspended to enforce security policies.

Figure 21-3 illustrates two different use cases of Directory Integrator for identity data management in Identity Manager.

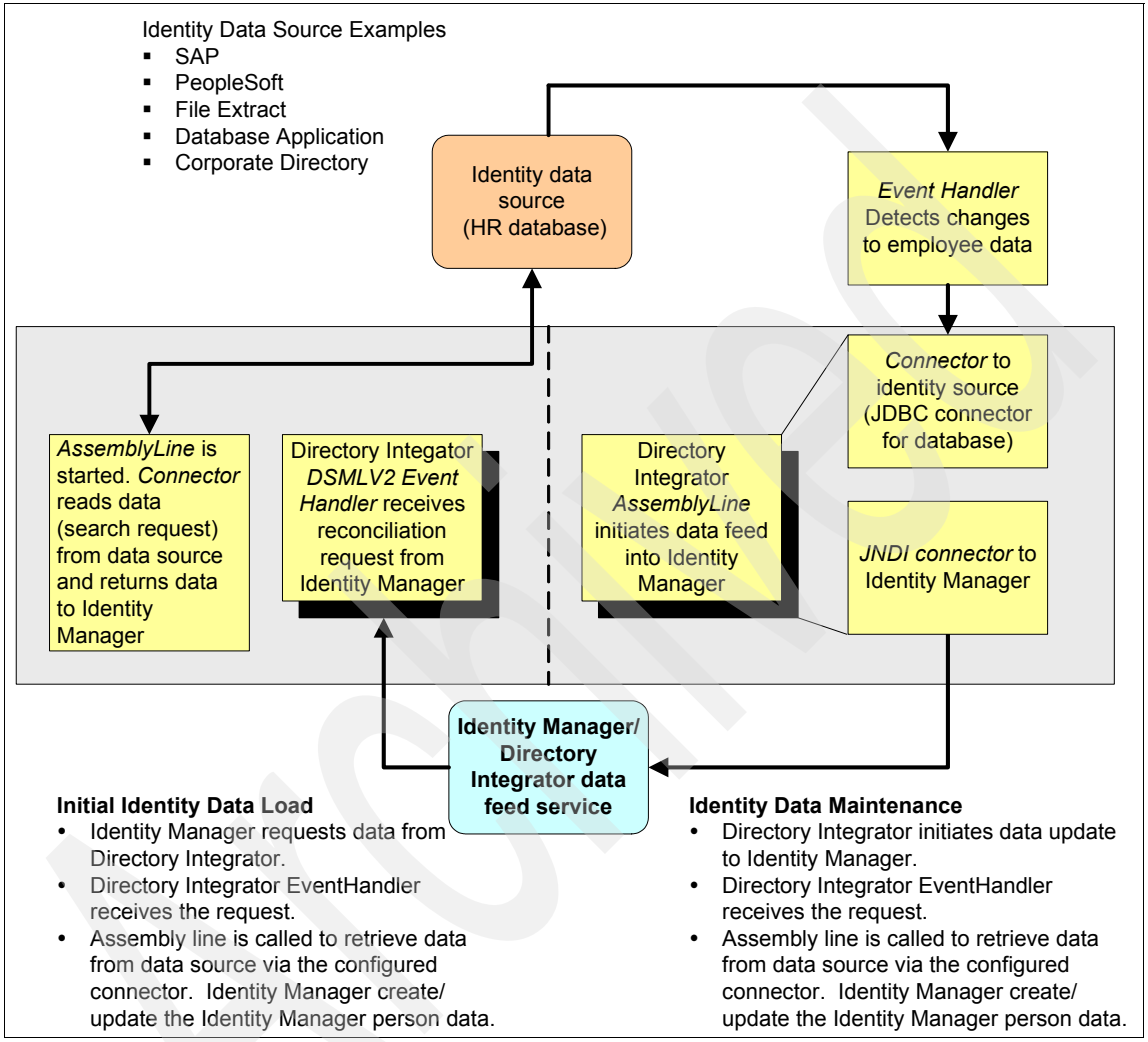


Figure 21-3 Accelerating Identity Manager deployments

Initial data load

In Tivoli Identity Manager, integration with Tivoli Directory Integrator is built into the Identity Manager server and into Directory Integrator components. One of these components is the DSMLv2 EventHandler in Directory Integrator that enables Identity Manager to make reconciliation requests to Directory Integrator, where Directory Integrator is configured to read data from a connected data

source. DSMLv2 is a standard that describes directory operations in an XML format. This is one of the quickest ways to implement a bulk data load of user information into Identity Manager. Another example to use DSMLv2 is for Tivoli Directory Integrator to push data into Tivoli Identity Manager using a DSMLv2 connector to a Director Integrator service definition in Identity Manager.

The integration task is made by Directory Integrator that is flexible in that the connected source can be another directory, an HR database, a flat file, or a DSML(v1) file, to name a few options.

The data flow for Identity Manager pulling data from Director Integrator is as follows:

1. A service is created in Identity Manager to connect to the URL for the Directory Integrator EventHandler.
2. When the reconciliation request is scheduled, Identity Manager contacts the Directory Integrator EventHandler with a DSMLv2 search request.
3. The Directory Integrator EventHandler is configured to start an AssemblyLine for the search request.
4. The AssemblyLine has a connector that iterates through the desired data source. Additional attributes and logic can be added in the connector's hooks or AssemblyLine flow. An example might be assigning a role or alias.
5. Data is returned to Identity Manager and person entries are created or modified.

The data flow for Director Integrator pushing data into Identity Manager is as follows:

1. A service is created in Identity Manager so a Directory Integrator DSMLv2 connector can connect to an Identity Manager URL.
2. Some event starts the Directory Integrator AssemblyLine.
3. The AssemblyLine has a connector that iterates through the desired data source. Additional attributes and logic can be added in the connector's hooks or AssemblyLine flow. An example might be assigning a role or alias.
4. Data is pushed into Identity Manager through a DSMLv2 over an HTTPS transaction and person entries are created or modified.

Identity data maintenance

After the person information has been populated into Identity Manager, how do you keep that data updated? Directory Integrator can also contact Identity Manager directly and send updates to Identity Manager using a DSMLv2 JNDI connector. Again, DSMLv2 specifies directory operations such as add, modify, delete, and search.

An example of a dataflow might be:

1. The Human Resources database is configured to detect changes to certain pieces of person information via a trigger or stored procedure. An AssemblyLine is configured using the database change connector in Directory Integrator. Not all changes to person data may have to be passed to Identity Manager, and Directory Integrator would be configured to only work with the attributes of interest to the Identity Manager information store.
2. The output connector would be configured to use the DSMLv2 JNDI connector to Identity Manager. This connector would be configured in the appropriate mode (for example, whether you are sending changes or deletions). In the case where a person might be terminated in the HR database, you may not want to delete them in the Identity Manager application but instead, mark their accounts as suspended. This is done straightforward with Directory Integrator as logic can be added to the AssemblyLine to set certain attributes to be a value based on the value of another attribute. For example, if employee status=TERM, you may want to change Identity Manager person status attribute to a value which would suspend the person and all of their accounts.

Identity Manager data maintenance

Another interesting use of Directory Integrator is to keep Identity Manager information synchronized. Suppose for a given company that the Active Directory account is the authoritative account for a person's e-mail address. If a user's e-mail address changes, it will be updated eventually in the HR database, but this process may take days or is a manual process. Changes made to Active Directory are reflected in the person's Identity Manager Active Directory service account information. This information is updated upon a reconciliation, which may happen on a daily (or more or less frequent) basis.

Using the LDAP changelog connector, these changes can be used to update the person information branch.

This is important because Identity Manager uses the person's e-mail address for notifications and possibly for password-reset information. Using Directory Integrator in this capacity helps to ensure that a user's e-mail address is always updated, based on this scenario's requirements.

21.5.3 Multiple directories and Tivoli Access Manager

Tivoli Access Manager requires a repository for storing identity and credential information. There are several choices as to which repository can be used (IBM Tivoli Directory Server, Active Directory, Novell e-Directory, to name a few). However, for business or technical reasons, the company may choose to not use the directory that is the authoritative source for the users, but instead build a new

directory and keep it and the authoritative directory synchronized. This separation of enterprise data and application data may also provide a more secure solution.

In this scenario a company is using Access Manager for Web access control and single sign-on. However, the authoritative source for user identity information is in Active Directory. But for technical reasons (the Active Directory is configured into multiple domains) and policy reasons (no changes are allowed to the Active Directory schema) a separate directory is required for Tivoli Access Manager, and IBM Directory Server will be used as the repository for Access Manager.

To keep the identity information synchronized, IBM Directory Integrator is used. A simple AssemblyLine is built using two connectors:

- Active Directory Changelog Connector to detect and read changes.
- LDAP Connector to update Directory Server (adds/updates/deletes) with user information including selected attributes, and to create and update the Tivoli Access Manager through the use of the Access Manager Java Admin API.

Figure 21-4 on page 645 depicts the solution overview with the data flow as follows:

1. A change is made to Active Directory. This can be add, modify, or delete of an entry.
2. A Directory Integrator AssemblyLine is running with an Active Directory Changelog Connector. The Active Directory Changelog Connector is a specialized instance of the LDAP Connector. This connector contains logic to poll Active Directory for changed objects. The Active Directory Changelog Connector adds the changeType attribute to every Entry returned. The possible values of the changeType attribute are add, modify and delete. These are used to represent new, changed, and deleted objects, respectively.
3. The entry information of the change is passed to the next connector in the AssemblyLine, and that connector updates Directory Server. The connector implements two important functions:
   a. The connector updates or adds attribute information to the person used by Tivoli Access Manager.
   b. Use the Tivoli Access Manager Java Administration API to create and modify the entry information that is used by Access Manager.

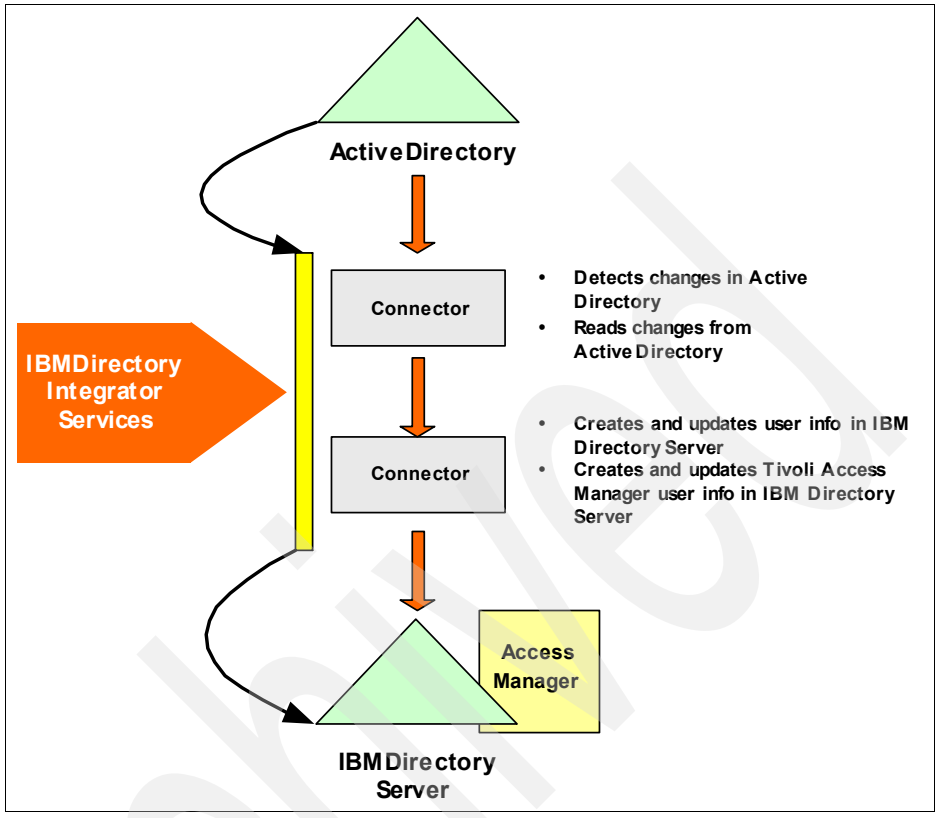


Figure 21-4 Multiple directories and Tivoli Access Manager

From this example you can see the power of Directory Integrator by its ability to incorporate Java functions directly within the logic flow of the solution. Most applications today have some kind of API interface available to extend the functionality of the product. Directory Integrator can fully exploit these APIs to provide a deeper and more precise level of integration where none may exist.

21.5.4 Password synchronization services

It is estimated that a company's Help Desk costs for password management can reach hundreds of thousands of dollars a year. For some companies, most of the ROI can be achieved with an identity management solution that focuses on password management alone. Savings can come in the form of reduced number of calls, reduced staff, and productivity savings.

The types of services that bring these savings to an organization are self-service password resets and password synchronization. Password synchronization is the

ability to detect a password change on one system and propagate that change to other password-protected accounts associated with the user.

IBM Tivoli Directory Integrator can detect password changes through the use of plug-ins. Directory Integrator can detect password creation and replacement for IBM Directory Server and IBM Lotus Domino HTTP passwords. The Password Synchronizer for Windows intercepts changes in user accounts for this platform.

Another feature is the ability to provide password interception for RACF. When RACF is configured to encrypt and store password changes into z/OS LDAP store for RACF, the Directory Integrator z/OS LDAP changelog EventHandler can detect the change and provide synchronization with other identity stores.

IBM Tivoli Identity Manager also has the ability to detect password changes from Windows systems, RACF when RACF is configured to encrypt and store password changes into z/OS LDAP store for RACF, and Tivoli Access Manager for e-business WebSeal. In Identity Manager, the server provides a standard interface that accepts password synchronization requests from other programs such as Directory Integrator.

Our company scenario has a simple single requirement: when a user changes a Tivoli Directory Server password, synchronize that password with all of the user's other accounts.

Identity Manager alone would not be able to provide the solution as there is no capability to synch Tivoli Directory Server passwords. Directory Integrator alone could sync the passwords, but there would be no knowledge of other accounts the person may have. By combining the two products, we can build a complete end-to-end solution to meet the company's requirement.

The diagram in Figure 21-5 on page 647 provides an overview of the solution.

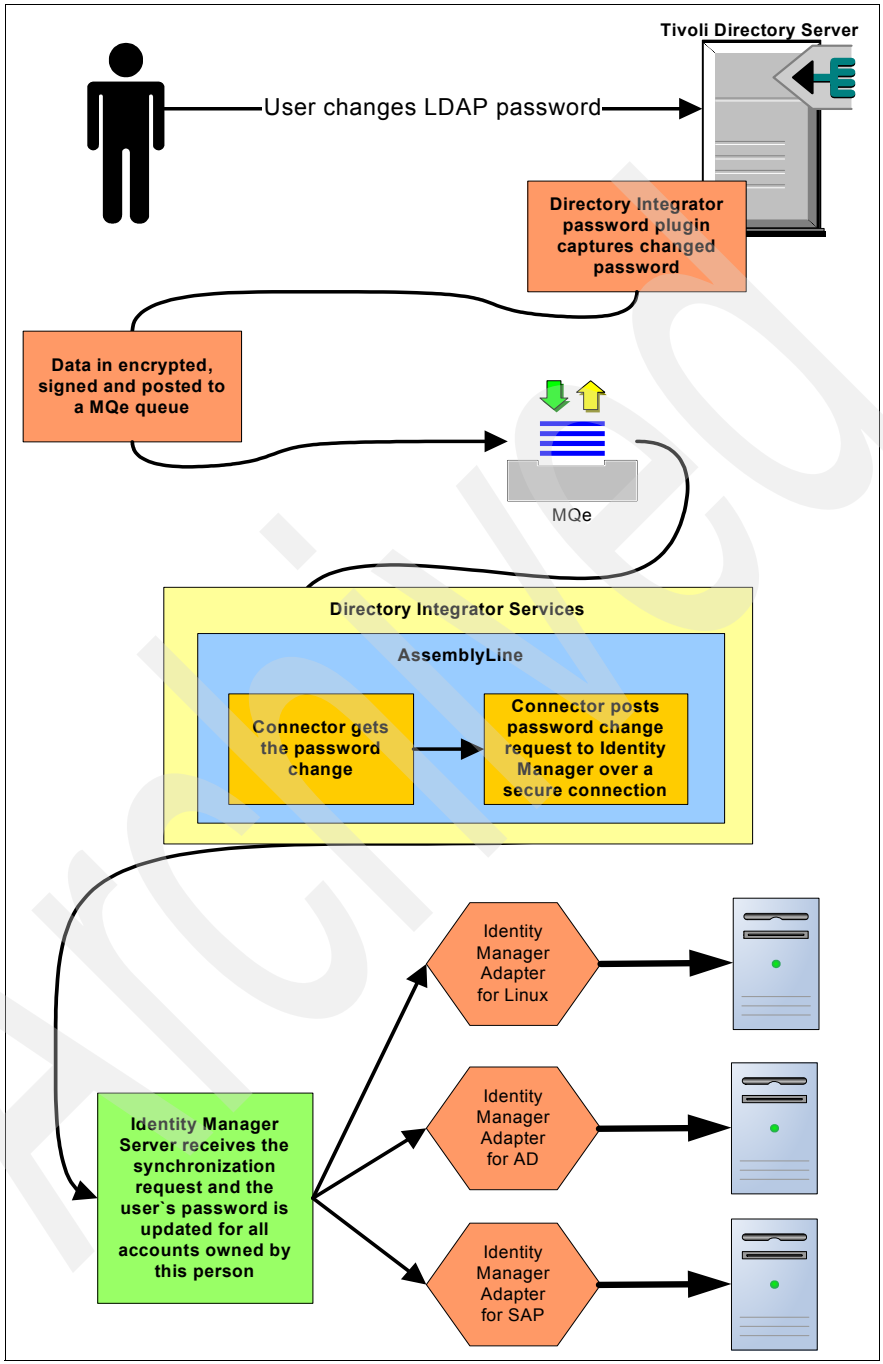


Figure 21-5 Password synchronization services

The event flow is as follows:

1. The user changes a password in Tivoli Directory Server. This change is captured by the LDAP changelog plug-in, and the password is encrypted, signed and posted in am MQe queue.
2. The EventHandler calls a Directory Integrator AssemblyLine that is composed of two connectors.
3. The first connector securely retrieves the encrypted password attribute and decrypts it with the supplied API in Directory Integrator.
4. The next connector builds the XML-structured request to Identity Manager to request the change. The connector does an HTTP post to the Identity Manager password-synch servlet to make the password change request.
5. Identity Manager receives the password synch request and checks to see what other accounts are eligible for password synchronization. Password change requests are then automatically initiated for the eligible systems (for example, Linux, SAP or Active Directory).

As you can see from this example, Identity Manager and Directory Integrator complement each other in functionality and provide the synergy to implement complex solutions quickly.

21.5.5 Migration services

Tivoli Directory Integrator is a powerful tool that can be used for many integrations, including migration services. The following scenario shows an Active Directory to Samba/Ldap migration using Tivoli Directory Server as the LDAP Server for Samba.

Figure 21-6 on page 649 shows how migration requests are generated.

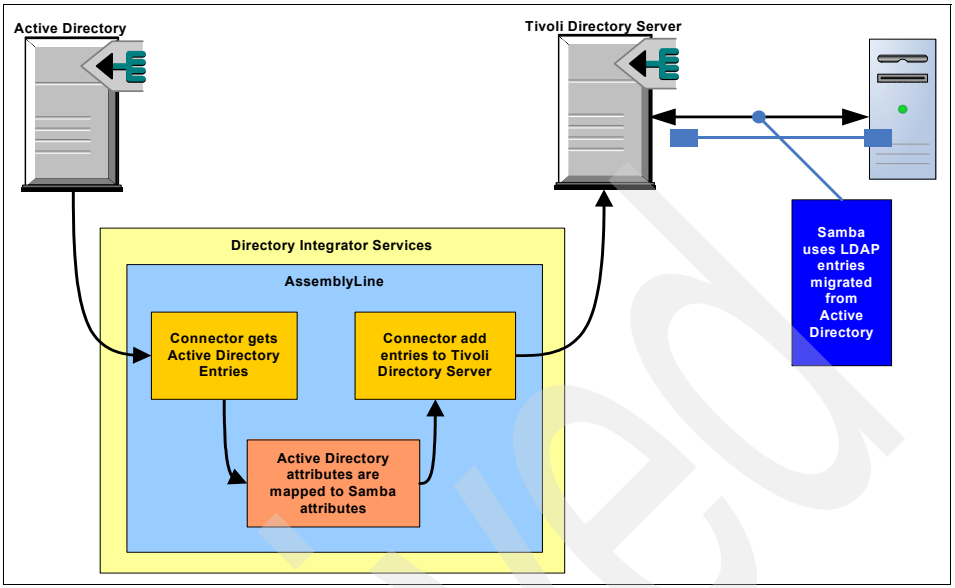


Figure 21-6 Active Directory to Samba/LDAP migration

21.5.6 Enabling Web portals

Most companies today are implementing portal applications. Portal solutions typically require a use repository to provide authentication and authorization information. One of the biggest challenges a company faces when implementing the portal is populating the repository for user access. A company may already have information for authentication and authorization in a single repository or multiple repositories.

The following scenario describes how Directory Integrator can be used to assist in a portal implementation where the authentication data may be stored in one directory, such as Active Directory, but the authorization and personalization data is stored in another directory (for example, IBM Directory Server).

WebSphere Application Server and WebSphere Portal are limited to using a single copy of a directory for authentication and authorization, so a solution is needed to work around this limitation by putting the authentication information into IBM Directory Server and then using one store only. We face an additional challenge as passwords are not transferable because they are one-way hashed. Even if password synchronization were available, it would solve only the problem of catching the password when it was changed. How can the user ID and password information get put into one directory to allow for authentication and authorization to happen for WebSphere Portal?

An approach would be to have WebSphere Application Server and WebSphere Portal use a plug-in to Directory Integrator to verify the authentication process and then use the IBM Directory Server for the authorization and personalization information.

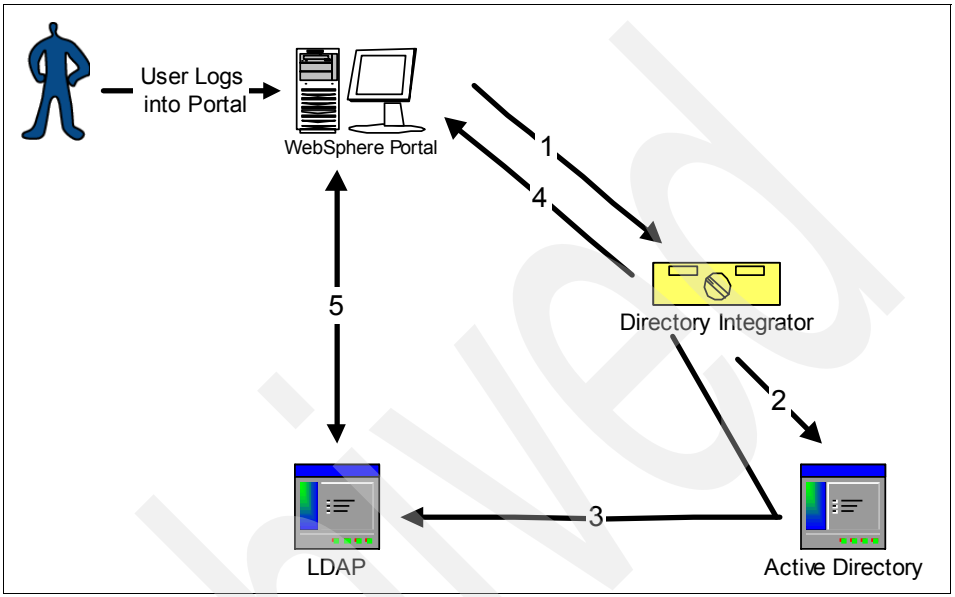


Figure 21-7 Enabling Web portals

Conceptually, the solution flow depicted in Figure 21-7 works as follows:

1. A user logs into the portal and provides an Active Directory user ID and password.
2. Instead of forwarding the request to WebSphere Application Server for authentication, the login page redirects the user ID and password information to Directory Integrator. Directory Integrator starts an AssemblyLine with connectors to Active Directory and IBM Directory Server.
3. Directory Integrator validates the user ID and password by performing a bind to the user in the Active Directory. If the bind is successful, the password proves to be valid.
4. The Active Directory user ID and password information is added to the corresponding user information in IBM Directory Server.
5. Directory Integrator returns a response back to WebSphere Portal indicating success or failure.
6. If successful, control is returned to the standard user authentication process within WebSphere.

During further processing, WebSphere Portal performs regular user authentication and authorization against the IBM Directory Server LDAP as configured in the security settings. No further interaction with Directory Integrator is required as the user ID and password are now stored in the main LDAP directory.

From this example, the flexibility of Directory Integrator is illustrated by the ability to provide *just-in-time* processing that overcomes some of the technical challenges presented in this company's situation.

21.6 Conclusion

Synchronizing security-related information within an enterprise can be somewhat of a challenge. With IBM Tivoli Identity Manager, Directory Integrator, and Directory Server you have a set of viable tools at your disposal to integrate with almost every data repository available in the market today.

In addition to functionality, the scenarios provide a good overview of how you can tackle synchronization challenges with adequate resources in an always-tight budgetary environment.

For more detailed information about Tivoli Directory Integrator, refer to the *Robust Data Synchronization with IBM Tivoli Directory Integrator*, SG24-6164.

Part 4

Managing federations

In Part 4 we take a look into the rapidly expanding world of federated identity management, Web services security and provisioning by introducing the IBM Tivoli Federated Identity Manager solution.

22

Business context for identity federation

The increasing ease and prominence of electronic commerce encouraged the trend for organizations to share and consume business services in order to meet their business needs. An organization may have relationships with business partners, customers, suppliers, and service provides. This same organization may also provide services to other organizations as well. All of these relationships involve various forms of trust between the organizations. These include electronic trust as well as traditional contractual obligations that define trust in terms of legal parameters around the business relationship. Identity federation uses the electronic trust relationships to allow organizations to securely execute business services across the organizational boundaries.

22.1 The business context

To describe the business context we look at some common business drivers for federated identity management. An organization is not an isolated entity. It will agree to provide or purchase services to other firms. Business partnerships will require interfacing IT systems between both companies to collaborate. Suppliers may be required to interface with the organizations system as part of contracts. The organization itself may also be required by customers to interface with the customer systems as well. We start to see a service oriented look to these arrangements. Each relationship represents a provided, delivered or shared service between different organizations.

These relationships and services include areas that require the exchange of extremely private and sensitive information. A strategy that allows for real time dynamic changes to integrate the services and identities in these services is needed.

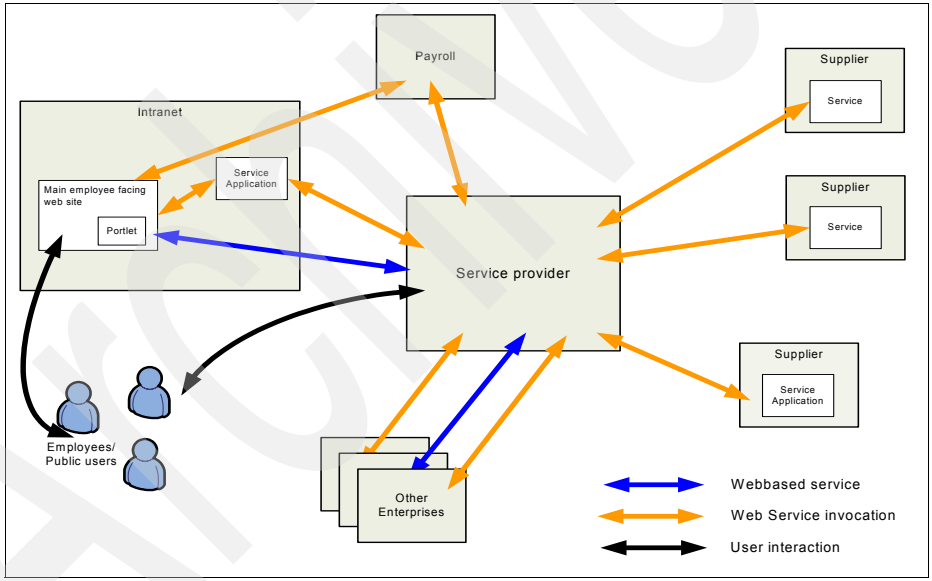


Figure 22-1 Business relationships involve many organizations

Figure 22-1 illustrates some of the integration points that must be addressed for these services to support new or existing business processes. A company (intranet in Figure 22-1) *out-sources* the administration of its telecom and related services (to service provider) and also its payroll services (to Payroll, shown in Figure 22-1). Service provider in turn has similar relationships with other

customers, and with its own suppliers. Note that service provider also out sources its payroll activities to the Payroll entity.

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

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

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

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

The business context involves a changing dynamic business climate with shared services. An underlying concern is trust and assurance found in 22.3.1, “The relationship - trust and assurance” on page 662. There are also business models for federated identity, which are discussed in 22.2, “Business models for federated identity” on page 657. The role of identity management is discussed in 22.4, “The role of identity management” on page 664.

22.2 Business models for federated identity

The business context presented some of the business relationship challenges found today. The following are business models or areas, that show a need for federated identity management.

Mergers and acquisitions

In this scenario a company is implementing a growth strategy using mergers and acquisitions. The indicator of the success of the merger or the acquisition is predicated on how quickly the companies can knit together their IT infrastructures to target and cross-market to the new customer base. Identity

management is one of the most complex activities in such mergers. Rather than having to forklift all of the acquired users in the various systems, an integration strategy based on identity federation can simplify the user experience. The combined users of the merged customers can have access to the shared assets of the merged companies without impacting user experience, customer care, or the quality of support. Federating the identities between the merged companies provides a quick and seamless way to integrate the customers of the two companies to drive merged growth scenarios.

Collaboration between autonomous cross-business units

Many large companies have independent business units that want to directly maintain ownership and relationship with their users. This may be due to organizational structure, or to political, competitive, or regulatory reasons. A large global manufacturing company may be organized as independent companies with regional management consolidated in the Americas, Europe, Africa, Middle East and Asia. However, these business units may also need to have their users (employees) needing access to cross-business unit resources. For example, employees in Asia need access to ordering and parts information in other regions. Federated identity management enables business units to retain autonomy and control of their users, yet have a flexible way to federate data to cross-business unit resources.

Customer acquisition strategy via partnerships

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

Employee access to outsourced provider services

Employee self-service is a major initiative for many companies looking to reduce user provisioning or user care costs. Most organizations outsource non-critical competencies to third-party providers. The services that are being outsourced include human resources, employee savings plan, healthcare, payroll, travel and procurement services. Using the corporate intranet portal to connect the

employees directly with these external service providers enables the organization only the administration of these outsourced services. Organizations outsource these services to reduce these service administration costs. However, the inability to directly connect the employee to these service providers means that the organization is now required to support and maintain staging systems (help desk) for employee enrollment to savings plans (401K or super-annuation), healthcare or payroll. Employers spend significant amount of plan administration costs in employee 401K administration, employee stock options, employee healthcare and travel. These services are typically outsourced to various outside providers. However, the company stills ends up manually administering these plans or having to staff customer care personnel for employee management.

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

Service provider automation with B2B clients

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

Portal-based integration of software-as-services

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

Government collaboration

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

Improved corporate governance

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

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

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

- Organizations cannot account for access rights granted in their internal systems for third-party users; there is no proof of whether a third-party user actually exists or even needs access.

- No accountability on why a third-party user was granted access in the first place; failure to demonstrate and document the business reason for the request and which company officer approved the request.
- No procedures in place to delete entitlements or purge user access rights belonging to third parties and their users. This results in users accumulating access rights far beyond what they were originally authorized.
- No procedures in place to de-provision user accounts when users turn over. This issue is magnified when dealing with third parties when the company does not control the third-party user and no process typically exists by which third-party companies will notify of user turnover.
- No way to re-certify third-party user access. Does this third-party user still need access beyond three months or six months? Why do they still need access?
- No way to audit request for third-party access. Most companies are not able to audit third-party user access in a centralized fashion because there is no one single tool that is being used to grant third-party access.

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

22.3 Federated identity

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

Federated identity management is based on the business agreements, technical agreements, and policy agreements that allow companies to interoperate based on shared identity management, depicted in Figure 22-2. This helps companies to lower their overall identity management costs and provide an improved user experience. It leverages the concept of a portable identity to simplify the administration of users and to manage security and trust in a federated business relationship. The simplification of the administration and the lifecycle management in a federation leads to the following value proposition:

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

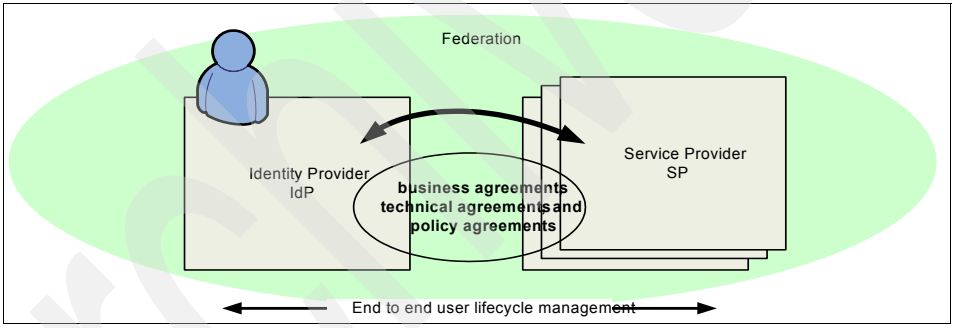


Figure 22-2 Federated identity management

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

22.3.1 The relationship - trust and assurance

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

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

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

- Is there a legitimate business with the stated name?
- Is this the party making the request?
- Is the specific employee making the request authorized?

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

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

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

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

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

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

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

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

22.4 The role of identity management

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

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

User lifecycle management of identities

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

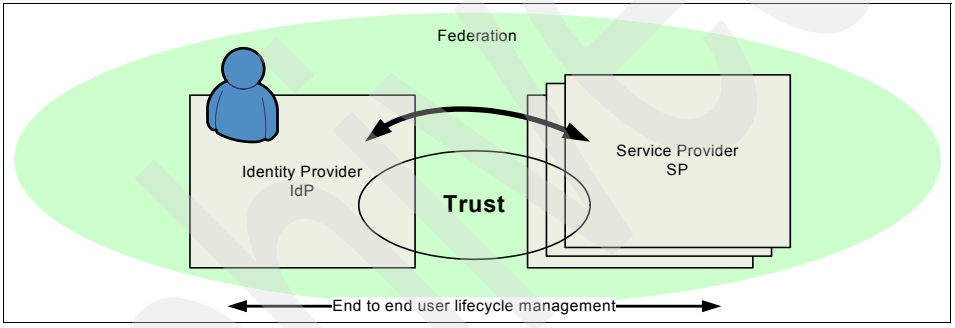


Figure 22-3 End-to-end user lifecycle management

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

Federated identity

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

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

Partnership-based solutions

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

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

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

Security characteristics

In a B2C or B2E[1] environment where consumers and employees communicate with one company as a focal point for multiple business partners, it is important to secure access to all involved parties. In B2B environments business partners and applications must also be used in a secure and reliable way.

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

In such a dynamic environment, *trust relationships* between business partners are essential. Traditionally, IT infrastructures have dealt almost exclusively with their own environments—not necessarily reflecting the needs of *interoperation*

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

and *integration* with other parties. In an truly dynamic business environment all parties must interact seamlessly to meet the requirements of a dynamic business. Figure 22-4 highlights a security triangle that these three elements form.

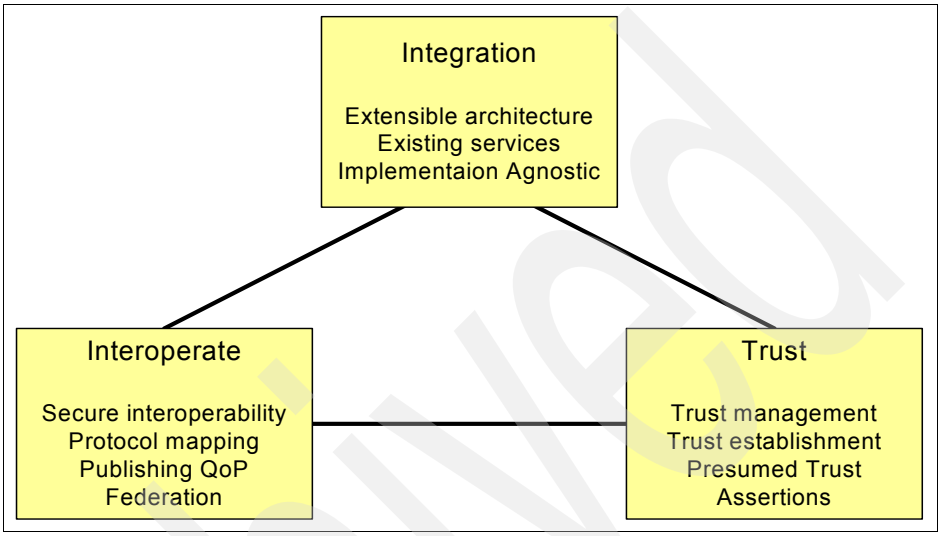


Figure 22-4 Security triangle: Trust - Interoperation - Integration

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

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

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

22.4.1 Dealing with identities

Figure 22-5 shows a typical business that deals with at least three major clusters of identities.

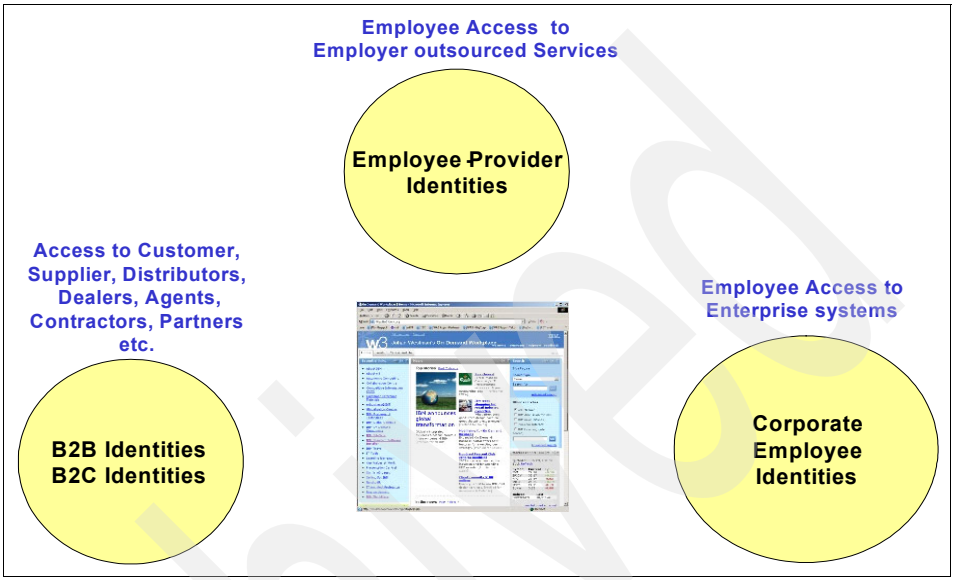


Figure 22-5 Dealing with identities - A corporate view

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

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

Corporate identities

A corporation typically has a number of systems and applications where their users need identities. The user needs to sign on to her workstation, possibly again to her corporate intranet, and may need to sign on again to the back-end

systems. These sign-ins may need multiple identities, which need to be managed as well as the user needs to remember all of them.

- Network identities (Remote Access, VPN or Wi-Fi Accounts) to enable users to access the network
- Desktop identities to sign on to the workstation (Windows credentials)
- Corporate e-mail and white pages accounts
- Existing accounts for mainframe accounts
- HR accounts (PeopleSoft, SAP, Oracle)
- Supply Chain/CRM accounts (SAP, Siebel, etc.)
- Identities that are managed in middleware and database solutions (Oracle accounts, WebSphere accounts, Portal accounts, and so on).

Employee to employer-outsourced provider identities

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

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

Business to consumer identities

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

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

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

22.4.2 User lifecycle management

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

- Improve and increase confidence in business transactions

  Identity is the basis of security; poor identity management means weak security

- Lower administrative cost

  Soaring costs with account information administration and password administration, user registration, and help desk support

- Risk, compliance, security exposure
  - Business, legal and privacy issues with user data access (for example, Sarbanes-Oxley, HIPAA, Graham-Leach-Bliley, CA SB1386)
  - Issues with unauthorized access from users
  - Audit failures due to inactive user account exposure
  - Identity and password theft
- Poor market reach
  - No standard mechanism to trust identities from M&A, business partners, and third parties
  - High cost of integration applications that deal with identities

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

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

User provisioning and account management costs

The cost of provisioning users with account data is one of the more expensive and manual activities that take people, time, and a significant IT budget. While automated user provisioning tools automate (synchronize) many aspects of user

provisioning, the fundamental issue still remains that a company takes on *user ownership costs* when provisioning account data. While a company may need to take on this user ownership cost for employees, this approach may not be correct when dealing with external identities that are currently being provisioned in the internal systems.

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

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

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

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

User registration and enrollment costs

There are costs associated with registering and enrolling a new user in the systems. User registration and enrollment costs accrue from the administrative processes that need to be deployed across the Interactive Voice Recognition

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

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

Password management costs

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

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

22.4.3 Inter-enterprise application to application integration

As mentioned in 22.1, “The business context” on page 656 businesses are interacting with services, and the *Service Oriented Architecture*, SOA, is a key strategy that the market is adopting to support the businesses drive towards becoming on demand businesses. Here we focus on the application to application integration challenges within SOA. SOA as mentioned earlier spans the own private business into new inter-enterprise interactions.

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

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

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

The Enterprise Service Bus

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

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

On demand applications are business services built from services that provide a set of capabilities that are worth advertising for use by other services. Typically, a business service relies on many other services in its implementation. Services interact via the Enterprise Service Bus, which facilitates mediated interactions.

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

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

22.4.4 Open standards

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

See more detailed description of open standards relating to federated identity management in 23.3, “FIM standards and efforts” on page 698.

F-SSO standards

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

Web services

Web services have emerged as the most promising development to address cross-enterprise, cross-platform, and cross-vendor business integration issues. Web services is a family of emerging technologies that enable easy interoperability of programmed information technology (IT) services and integration of applications into a company's broader business processes. Web services technology enables companies to describe available services and provide access to those services over standard Web protocols and communications boundaries.

Web services security specifications

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

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

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

Web services provisioning specification

WS-Provisioning is a specification authored by IBM to provide a Web service interface to communicate provisioning requests and responses. It includes operations for adding, modifying, deleting, and querying provisioning data. It also specifies a notification interface for subscribing to provisioning events. Provisioning data is described using XML and other types of schema. This facilitates the translation of data between different provisioning systems.

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

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

22.5 Conclusion

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

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

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

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

By choosing to operate in business federations, companies do the following:

- Reduce identity and security management costs through linkage and reuse between companies. Companies no longer need to separately manage users or identities that are not under their control, reducing identity lifecycle management costs.
- Achieve order of magnitude increases in efficiency through reuse of security infrastructure and end-to-end business process integration.
- Deliver simplified and trusted user experience with single registration, single sign-on because users can navigate easily between Web sites with a single

identity and explicitly control release of their personal data. Implement business strategies that drive organic market and customer growth by eliminating the friction caused by incompatible identity and security management between companies.

The IBM federated identity management solution delivers concurrent support for key identity management specifications, such as *Liberty*, *WS-Federation* and *SAML*. The IBM federated identity is built on the trust foundation of the WS-Security family of specifications. Integration of federated identity management capabilities with IBM middleware solutions, such as WebSphere enables application platforms to be integrated using industry standards.

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

23

Federation concepts

A federation is a group of two or more trusted business partners with business and legal agreements, including liability restrictions placed on the business partners. Participation in a federation allows a user from one federation business partner to seamlessly access resources of another business partner in a secure and trustworthy manner, be it directly using a Web browser or accessing a local application integrated to another business partner's application. This allows end users to easily accomplish the tasks they need to complete cross-company business transactions. This in turn promotes cross-company business in a loosely coupled environment.

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

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

The base for discussing how to architect a solution will be an example used throughout this book. Based on the example, the federated identity architecture is studied and terminology is explained along with the specifics of federated identity solutions.

23.1 Federation example

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

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

The involved businesses interact with each other creating a value net of services available to users, both public users or employees of a business, see Figure 23-1.

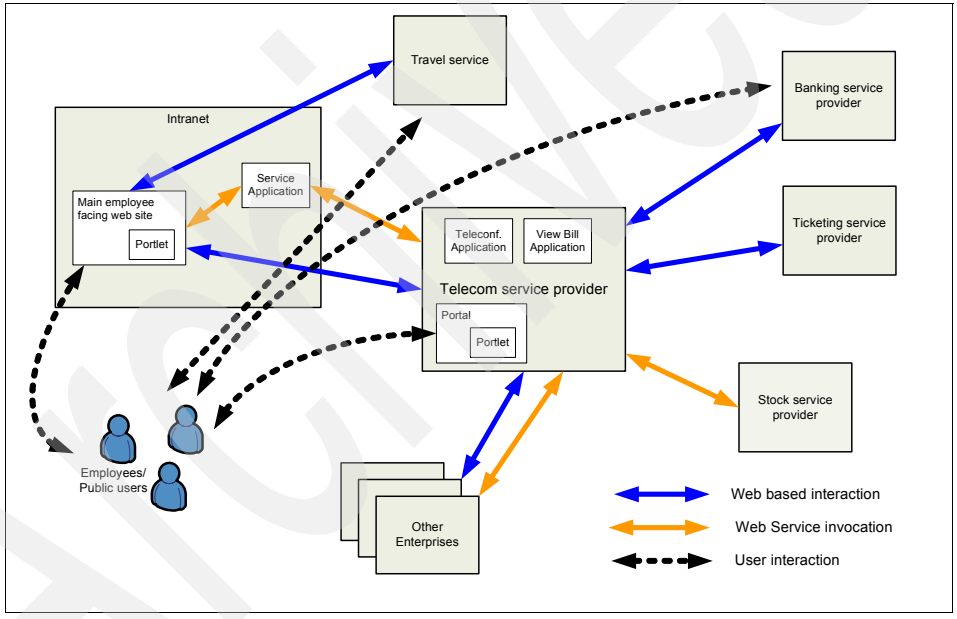


Figure 23-1 Federation example environment

BigCorp

BigCorp is a large company with many employees. As part of providing benefits for its employees, BigCorp provides (subsidizes) health care, retirement saving plans, and other employment related services (such as subsidized mobile phone accounts). As part of reducing its “employee costs”, BigCorp has out sourced these employee benefits to third-party benefit providers. As

	BigCorp is responsible for the management of its users, from account creation (initial hiring) to account deletion/inactivation (dismissal/retirement/other severance), it is nature for BigCorp One to continue to assume this functionality but to leverage this in its relationships with third-party benefit providers.
Employee One	Employee One is a typical BigCorp employee. He has access to the typical BigCorp provided (brokered) services. Employee One also leverages additional services brokered by BigCorp and provided by third-party providers, including travel services, a BigCorp sponsored mobile phone plan, participation in a stock plan and online banking.
RBTravel	RBTravel manages travel related services for other businesses, allowing them to order and pay for flights, trains, car rental, hotels and much more. RBTravel have agreements with the businesses using there service to allow anybody from their business who are directed to their Web site to allow them to automatically get an account.
RBTelco	RBTelco is a telecommunications service provider that offers telephony services and also has a portal where RBTelco users or business partner users can choose among offered services to which RBTelco will act as identity provider, offering SSO to the services. RBTelco also has services in their portal connecting to external service partners Web services and presenting them in the portal. RBTelco also acts as a service provider to large enterprises, like BigCorp.
RBBanking	RBBanking offers banking services to its own customers directly and also to RBTelco customers through their portal.
RBStocks	RBStocks offers a stock quote service. The service offers different service levels depending on the user of the service. The stock service is a Web service. RBTelco offers this stock service on their portal.

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

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

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

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

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

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

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

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

What is required is that RBTravel is able to participate in a runtime exchange of information with BigCorp which results in some assertion from BigCorp (note that this exchange of information requires no interaction with Mr. One). This assertion is then trusted by RBTravel and used to uniquely identify Mr. One based on an

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

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

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

Public users to the RBTelco portal would have an personal account. Public users who are customers of RBTelco would benefit from its partners service offerings presented by the portal. The services would allow for reduced sign-on allow the user only to log on to the RBTelco portal and then selecting that service by clicking on the link in the portal and then connecting to the offered services without having to log on again. One such service is the RBBanking, offering its customers access to their bank services through the RBTelco portal with reduced sign-on, in the same way as RBTravel offered its services to BigCorp employees.

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

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

23.2 Federated identity management architecture

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

few different solution areas shown in Figure 23-2. The solution areas are as follows:

- Web-based single sign-on—federated single sign-on referred to as F-SSO
- Application based Web services security—Secure Web services referred to as Web services security management
- Identity lifecycle—federated provisioning

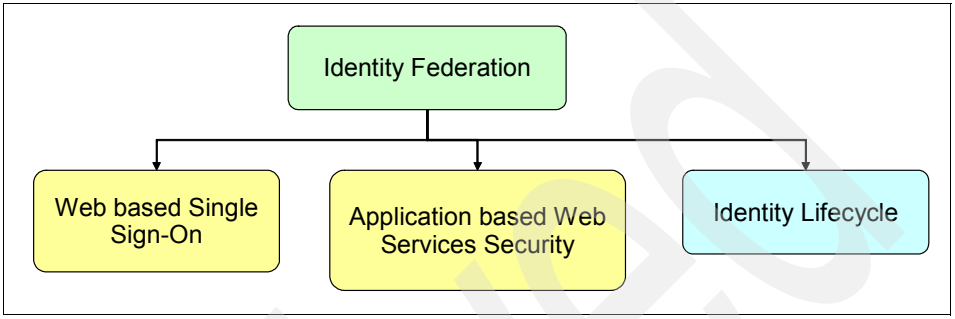


Figure 23-2 Federated identity management solution areas

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

In the sections following this one, the specifics of the three solution areas will be studied in a little more detail with regards to their functional specifics, 23.4, “Federated single sign-on” on page 705, 23.5, “Web services security management” on page 711 and 23.6, “Federated identity provisioning” on page 716.

Chapter 24, “Federated Identity Manager” on page 721 will do the same, but focuses on the IBM Tivoli Federated Identity product specific approach.

23.2.1 Background to federation

Federation solutions are successful when they allow customers, business partners and end users to integrate easily between the federation business partners without having to constantly manage security and identity in the process in a per relationship proprietary way. Unfortunately, current implementations for managing security and identity data often force users and businesses to manually manage access, trust, transport and identity attributes. Often this burden has a heavy impact on both ability to execute and growing administrative cost due to that each business has to administer a large and rapidly changing base of identities. Such a model is an impediment to the adoption of federations and is a pain point for both users and businesses, as we discussed in 22.2,

"Business models for federated identity" on page 657 and 22.3, "Federated identity" on page 661.

Federation technology is used to do the following:

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

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

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

One of the most pressing questions for an IT administrator is how to implement the technical policies and operational best practices; how to implement and enforce security and identity agreements, audit and privacy agreements, such that the federated relationships look like an extension of their existing identity management procedures.

23.2.2 Architecture overview

Federated relationships can be based on proprietary technologies that allow business partners to communicate and collaborate. In general, a proprietary approach is not scalable or maintainable across a large set of partners. For this reason, standards and specification based approaches are rapidly gaining in

popularity. Federations facilitate an integrated approach to business. Federations are entered in to facilitate two major types of functionality:

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

Both of these functionality types leverage the same basic functionality, namely both require a *trust infrastructure*. The trust infrastructure provides the technical representation and implementation of the business and legal agreements between business partners, as shown in Figure 23-3. Both federated single sign-on and Web services security solutions are built on a trust infrastructure.

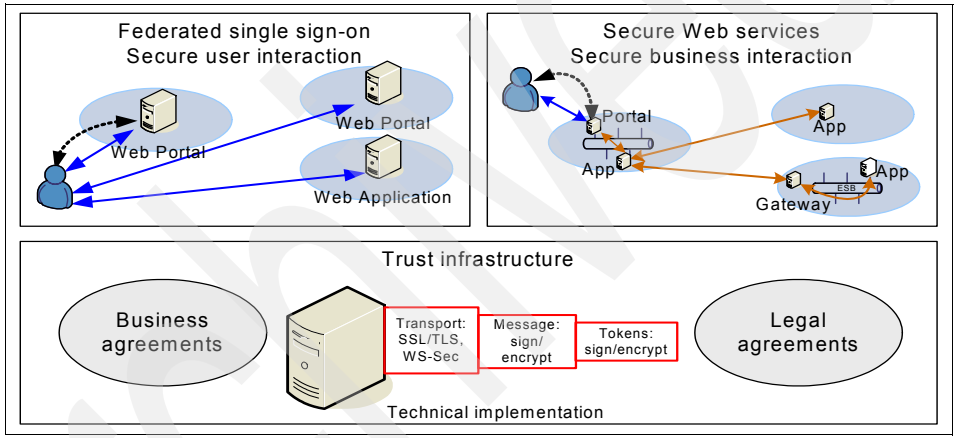


Figure 23-3 Base trust infrastructure for secure services

Federated identity management often refers to user-driven, browser-based interaction between organizations. This space is reference to as *federated single sign-on* (F-SSO) even though it has largely matured beyond just single sign-on functionality. Standards and specifications such as the SAML specification and WS-Federation and Liberty Alliance ID-FF specifications all now include an aspect of session lifecycle management (single sign-on and single logout) as well as single sign-on enablement through account linking. This comprehensive approach and enablement of a single sign-on environment is designed to ease the user experience and reduce the cost of management of these users. For example, previously a user had to establish an account, including username and password, at each business partner; the business partner in turn had to assume the cost of managing this user and their access to their system. Federation solutions ease this cost by reducing the amount of information that must be managed for each user and the overall cost of managing this information.

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

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

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

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

- The *roles* of identity and service provider: The definition who is the authoritative source of the user identity information
- Identity/attribute *mapping*: The definition of the attributes to be shared and the mapping of them in the business partner systems
- User account *management/provisioning*: The procedures for managing user identity data, agree what information can be shared, and what information is independently managed by users, and will the users be provisioned automatically to the new endpoint (a priori or runtime)
- Account *linking*: The procedures for managing the account linking, to agree on some common unique identifier for the user, which can be bounded with the internal, local user identity at the service provider. This step also involves the definition of the account de-linking/de-provisioning procedures
- *Trust*: The process of ensuring security for connections/transport, messages and tokens
- Selection of the federation protocol profile(s): The definition of the federation protocol profiles to be used between the two business partners

23.2.3 Roles

Within a federation, business partners play one of two roles: *Identity provider* and *service provider*. The identity provider (IdP) is the authoritative site responsible for authenticating an end user and asserting an identity for that user in a trusted fashion to trusted business partners. Those business partners who offer services but do not act as identity providers are known as service providers. See Figure 23-4. The identity provider takes on the bulk of the user's lifecycle management issues. The service provider (SP) relies on the IdP to assert information about a user, leaving the SP to manage only those user attributes that are relevant to the SP.

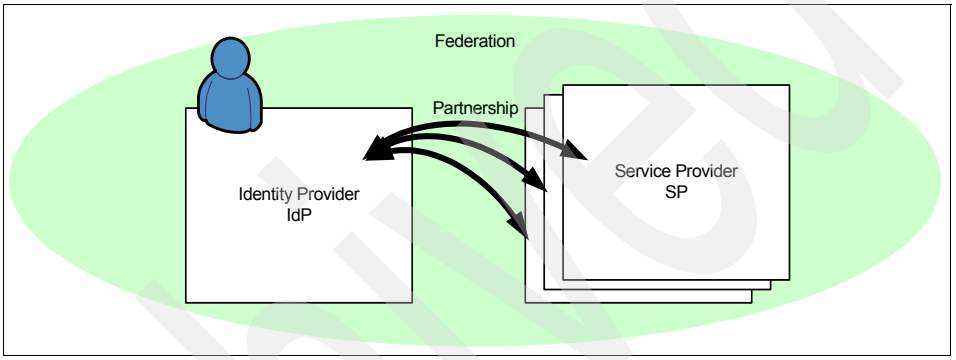


Figure 23-4 Identity provider and service provider in the federated model

Identity provider

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

In 23.1, “Federation example” on page 680, both BigCorp and RBTelco act as an identity providers. RBTelco is also a service provider.

To achieve the overall user lifecycle management required for a full federated identity management solution, the identity provider assumes the management of *user account creation*, *account provisioning*, *password management*, and *identity assertion*. The identity provider and service provider cooperate to

provide a rich user experience by leveraging distinct federated identity management profiles that together provide a seamless federation functionality for a user.

Service provider

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

23.2.4 Identity models

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

Shared

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

The fundamental question with respect to identity and attribute provisioning between business partners is what information can they share and what are the benefits of sharing? In an optimistic scenario an IdP and SP share every piece of information about the user as in Figure 23-5 on page 690.

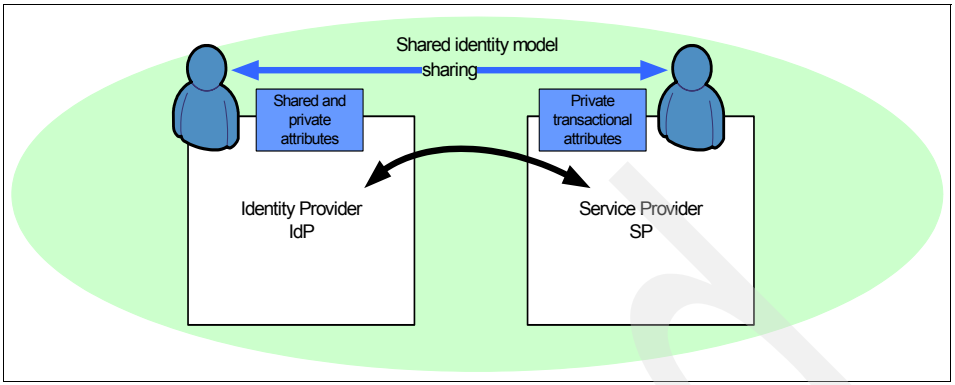


Figure 23-5 Shared identity model

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

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

- *Sharing transactional attributes* requires that the identity provider and service provider agree upon the roles and entitlements or groups that the user belongs to up front. This is a difficult proposition to implement, as two independent providers typically have different ways to group identities or manage role information. Rather than sharing transactional attributes, a provider may map their transactional attributes in a form that their business partner understands. In this approach identity metadata is maintained separately at both identity provider and service provider.
- *Sharing profile attributes* between identity provider and service provider is usually a function of user consent. This is more dictated by user preference and user privacy concerns. Sharing of attributes in many cases will require user consent (OPT in) and the ability to prove user consent. In a pragmatic sense, some attributes may be shared (such as e-mail address), while some attributes will not be shared. If attributes cannot be shared then the attributes need to be replicated between the identity provider and service provider. So if, for example, a user's home address is replicated, both business partners must independently manage this information. If the user moves, and one of

the business partners knows about the updated address, in a distinct identity model, the business partner cannot notify/provision this information to other business partners.

Provisioning plays a key role in determining all three of the above scenarios when the identity information cannot be shared between IdP and SP. This will be discussed in more detail in 23.2.6, “Trust” on page 695and 23.6, “Federated identity provisioning” on page 716.

Distinct

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

With a distinct identity data management model, identity data may be initially provisioned across business partners as part of the initial account setup, although it will be managed independently (outside the scope of a provisioning solution) after this as in Figure 23-6.

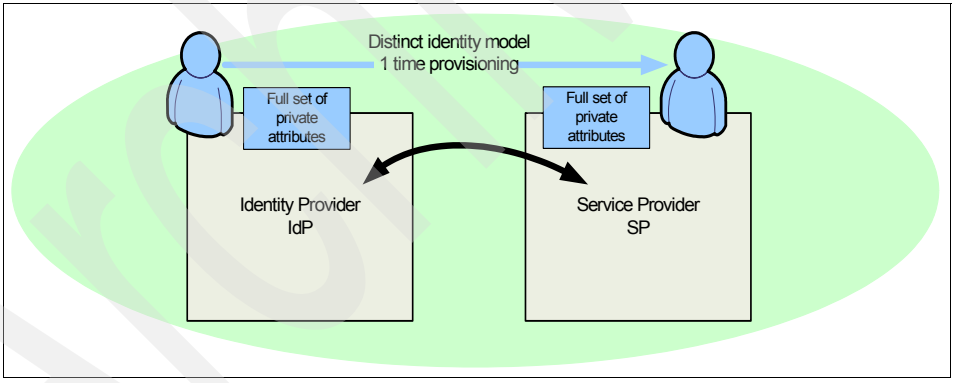


Figure 23-6 Distinct identity model

23.2.5 Identity attributes

In a federated model an identity provider and service provider need to agree on what information they can share with respect to a user identity and what information must be independently managed. This information is composed of classes of data that concern an identity:

- Authentication credentials
- Transaction attributes

- Profile attributes
- Provider-specific attributes

For each class of *identity data*, we can allow for a *shared* or *distinct* identity data management solution as shown in Figure 23-7. Thus when examining the provisioning requirements for a federated model, we evaluate the shared/distinct nature of each of the classes of identity data.

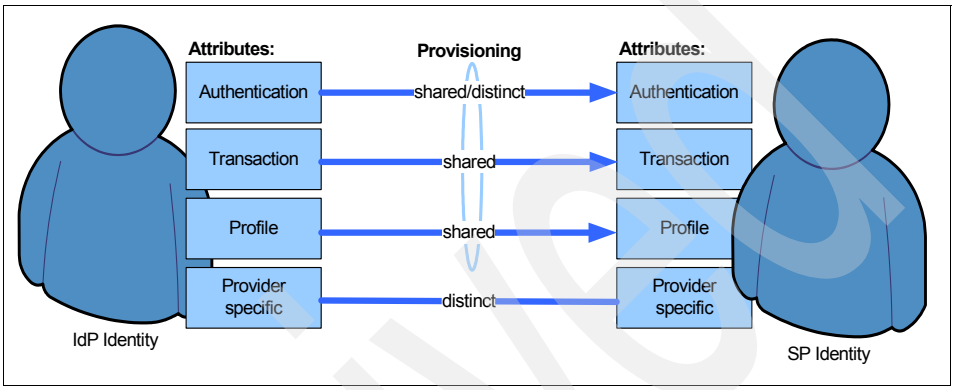


Figure 23-7 Shared and distinct identity data and attributes

Authentication credentials

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

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

Moving to a shared model for authentication credentials means that federation business partners are able to act as service providers and no longer have to manage the class of identity data, including authentication credentials. Provisioning solutions are used to tie the identity account management at an identity provider to that at a service provider.

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

In a shared identity model there is no need to provision authentication credentials. There is, however, a need to somehow establish a user's local identity and the common identifier used by the two business partners. This is handled through a provisioning solution. In general, a distinct identity account data model does not involve a provisioning solution. The user in this federation model has distinct identity accounts at both of these business partners, maintained and administered independently at both the identity provider and service provider. With a distinct identity data management model, identity data may be initially provisioned across business partners as part of the initial account setup, although it will be managed independently (outside the scope of a provisioning solution) after this.

There may be some cases where this is not true, for example, if the user does not already have a distinct, authenticable account at both the identity provider and the service provider. In this case, the identity provider may trigger a provisioning event at a business partner to create a local identity account and identity account data for a user. Part of this action may including establishing a common identifier used by the two business partners. As with the shared data approach, provisioning solutions when invoked within a distinct identity model may come in one of two flavors: Runtime (or just-in-time) and a priority provisioning, described in 23.6, “Federated identity provisioning” on page 716.

Transactional attributes

Transactional attributes include information that describes a user and his affiliations and entitlements. This information is bound to a user's identifier. This may include groups that the user belongs to or roles that he can assume. This data may also include additional identifiers (such as customer ID number, 401K account number, frequent flier status level, health care number, supplier ID, or billing or credit card number, etc.), specific organizational roles (such as HR

manager, stock broker, benefits administrator, primary care physician, executive, supervisor, travel exception approver, and so on).

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

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

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

Profile attributes

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

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

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

Based on Mr. One's travel account, the travel service will assign (and manage) Mr. One's frequent flier status (a transactional attribute). When attempting to book a flight Mr. Ones attributes will be used to assist him in booking the flight and also enable the ticket to by issued to his frequent flier card. When Mr. One attempts to book a trip, his travel class may be based on attributes with regards to his airline points or position at BigCorp. Once Mr. One has selected his desired travel and is about to book it, secondary evaluating of Mr. One's identity will be accomplished as part of the specification of Mr. One's billing address (to which the ticket confirmation information is to be sent).

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

Provider-specific attributes

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

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

23.2.6 Trust

Trust is a key capability for all three solution areas, and therefore a key area for FIM. Trust Services are also discussed in some detail in 24.2.4, “Trust services” on page 729.

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

To help ensure a desirable user experience, business partners within a federation need to communicate information about a user in a secure and trusted fashion. This is accomplished by leveraging a trust infrastructure.

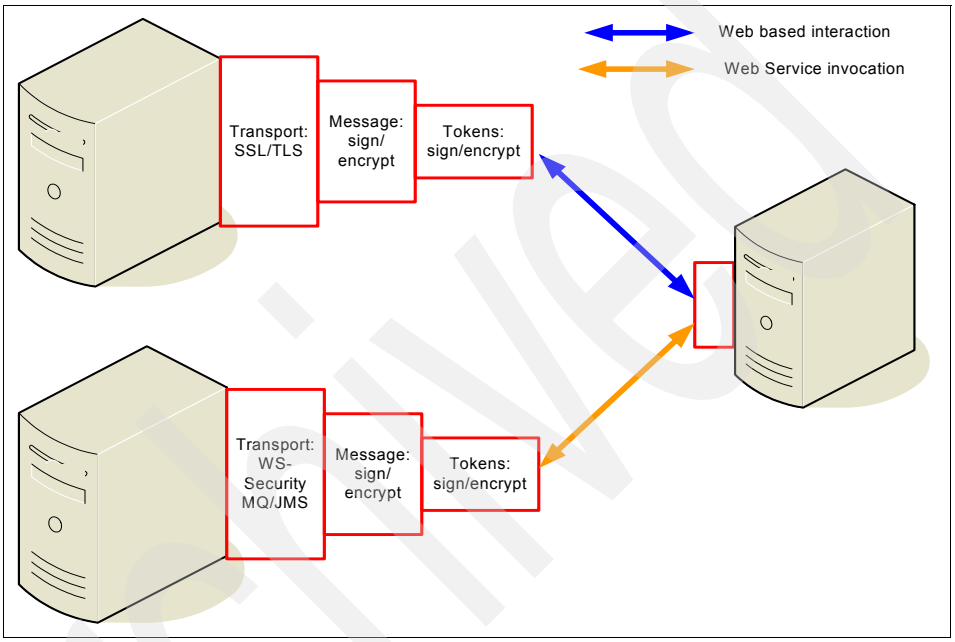


Figure 23-8 Layers of trust

A trust infrastructure enables the protection of a message at all levels, as shown in Figure 23-8:

Transport	Using SSL to protect user based FIM communications or WS-Security to protect application based FIM communications
Message	Using encryption and signing to provide confidentiality and integrity protection on messages within a FIM flow
Token	Using secure tokens to communicate information about a user as part of specific steps within a FIM flow

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

Transport

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

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

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

Message

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

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

Token

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

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

23.2.7 Federation protocol

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

The different standards and efforts in this space are discussed in 23.3, “FIM standards and efforts” on page 698. The different standards have different capabilities that govern the choice of protocol, made. Use the table in 23.3.9, “Selecting Federation standards” on page 703 to help select SSO protocol.

23.3 FIM standards and efforts

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

The first standards-based efforts in this space where by Internet (*Shibboleth*) and OASIS (*SAML*). Since then, federation efforts have been lead by the Liberty Alliance (*Liberty ID-FF*) and through the Web services work of IBM and Microsoft and partners (*WS-Federation*). Each of these efforts is introduced and briefly discussed in the following sections. The more recent Web services standards including WS-Security, WS-Trust and WS-Provisioning are presented as well.

23.3.1 SSL/TLS

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

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

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

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

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

23.3.2 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 SSTC (Security Services Technical Council). SAML has *two major components*: It describes *SAML assertions* used to transfer information within a single sign-on protocol and *SAML bindings and profiles* for a single sign-on protocol.

A SAML assertion is an XML-formatted token that is used to transfer user identity (and attribute) information from a user's identity provider to trusted service providers as part of the completion of a single sign-on request. A SAML assertion provides a vendor-neutral means of transferring information between federation business partners. As such, SAML assertions have a lot of traction in the overall federation space.

As a protocol, SAML has three versions, SAML 1.0, 1.1 and SAML 2.0. SAML 1.0 and SAML 1.1 (collectively, SAML 1.x) focus on single sign-on functionality. SAML 2.0 represents a major functional improvement over SAML 1.x.

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 lifecycle functionality than previous versions. Likewise, based on the Shibboleth input, SAML 2.0 has functionality that addresses some of the privacy concerns associated with a federated environment. SAML 2.0 is still largely in development with customer adoption/deployment expected to take off in mid-2006.

More information about the SAML specification is available from:

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

23.3.3 Shibboleth

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

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

23.3.4 Liberty

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

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

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

This is an excerpt from:

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

For more information about Liberty Alliance, see:

`http://www.projectliberty.org`

23.3.5 WS-Federation

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

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

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

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

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

The three specifications that make up WS-Federation are available for download from IBM DeveloperWorks at the following Web sites:

- WS-FED:

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

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

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

The logical architecture described in WS-Federation, together with the functionality described in the Web services security stack, supports both the active and passive client scenarios. The complete family of WS-Security specifications provides companies with a standards-based interoperable secure digital identity and trust platform for Web services- based architecture. Furthermore, these specifications promote reusability of existing IT security

investments, enabling companies to work with multiple security token types and multiple scenarios including vanilla browsers, enhanced browsers, active clients, and application-to-application connectivity.

There is also more information about WS-Federation in 24.3.5, “Federated single sign-on approaches” on page 753.

23.3.6 WS-Trust

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

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

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

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

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

Note that in July 2005 IBM and Microsoft announced that WS-Trust would be submitted to the OASIS standards organization. The announcement of Technical Committee (TC) formation is expected in September 2005, after which the normal OASIS process for standardization will begin.

23.3.7 WS-Security

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

The OASIS Security Services Technical Council, together with the OASIS Web services Security Technical Council, has defined a Web services Security SAML Token Profile. This describes how to bind a SAML assertion *in the context of WSS:SOAP Message Security, for securing SOAP message exchanges*.

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

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

There is also more information about WS-Security in 24.4.2, “WS-Security” on page 770.

23.3.8 WS-Provisioning

WS-Provisioning describes the APIs and schemas necessary to facilitate interoperability between provisioning systems and to allow software vendors to provide provisioning facilities in a consistent way. The specification addresses many of the problems faced by provisioning vendors in their use of existing protocols, commonly based on directory concepts, and confronts the challenges involved in provisioning Web services described using WSDL and XML Schema.

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

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

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

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

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

There is also more information about WS-Provisioning in 24.5, “Provisioning services” on page 775.

23.3.9 Selecting Federation standards

To help in selecting which F-SSO profile to use, see Table 23-1 on page 704. Table 23-1 on page 704 highlights some of the characteristics of each protocol: SAML 1.0 and 1.1 (OASIS standards), Liberty ID-FF 1.0, 1.1and 1.2 (Liberty Alliance published specifications), and WS-Federation (WS-Fed) Passive (IBM, Microsoft, RSA, VeriSign published specification).

Table 23-1 Characteristics per SSO protocol

Supported characteristic	SAML 1.0, 1.1	SAML 2.0	Liberty ID-FF 1.0, 1.1, 1.2	WS-Federation
PUSH SSO - Identity provider (IdP) initiated SSO	Yes[a]	Yes	No[b]	Yes
PULL SSO - Service provider (SP) initiated SSO	Yes	Yes	Yes	Yes
Front channel security token exchange	Yes	Yes	Yes	Yes
Back channel security token exchange	Yes	Yes	Yes	No[c]
Choice of security token type	No	No	No	Yes
Where are you from? (WAYF) support	N/A	Yes	Yes	Yes
Accounts at IdP and SP are required to initiate SSO	Yes[d]	Yes	Yes[d]	No
Accounts at IdP and SP are required to initiate account linking process	N/A	Yes	Yes	No
IdP-initiated account linking (federation) within SSO process	No	Yes	No	Yes
SP-initiated account linking (federation) within SSO process	No	Yes	Yes	Yes
Support for Single log out (SLO) or single logout	No	Yes	Yes	Yes
Create account on SP-side as part of IdP-initiated SSO or account linking - Just in time provisioning (JITP)	Yes	Yes	No	Yes
Account de-linking where user had pre-existing accounts before account linking	Yes	Yes	Yes	Yes
Account de-linking where user did not have pre-existing accounts before account linking	Yes[e,f]	Yes	Yes[e,f]	Yes[f]

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

c. The WS-Federation Passive scenario used to demonstrate interoperability employed a front-channel token exchange. Back-channel exchange can be supported using a direct trust server to trust server security token request, replacing the information passed in the front channel with an artifact-type security token.
d. The profiles for SAML and Liberty ID-FF explicitly require accounts at both the IdP and SP side as a prerequisite for SSO and account linking. A particular vendor implementation may not require this (see item 9 for more details).
e. This is somewhat out of the scope of SAML and Liberty ID-FF implementation as they both require that a user had accounts at both sides before the account linking process was initiated.
f. Assuming that the SP side account was created in response to runtime provisioning, this account must have been created in a manner that allows it to be converted from an SSO account to a direct-authentication account.

You can find more information about SAML, Liberty ID-FF, and WS-Federation in 24.3, “Federated single sign-on” on page 740.

23.4 Federated single sign-on

Federated single sign-on is the process by which a user authenticates to a federation business partner (an identity provider) and has the IdP assert a relevant identity (and attributes) to any/all required (and trusted business partner) service providers as part of the user's online federation experience. Global sign-on itself is provided by a federated single sign-on protocol (see 23.3.9, “Selecting Federation standards” on page 703). These protocols provide standard, interoperable means for multiple federation business partners to negotiate the presentation of credentials about a user from an identity provider to a (trusted) federation service provider.

In this section *federated single sign-on* functionality is discussed, this is also studied more in detail, out of an IBM Tivoli Federated Identity Management product point of view, in 24.3, “Federated single sign-on” on page 740.

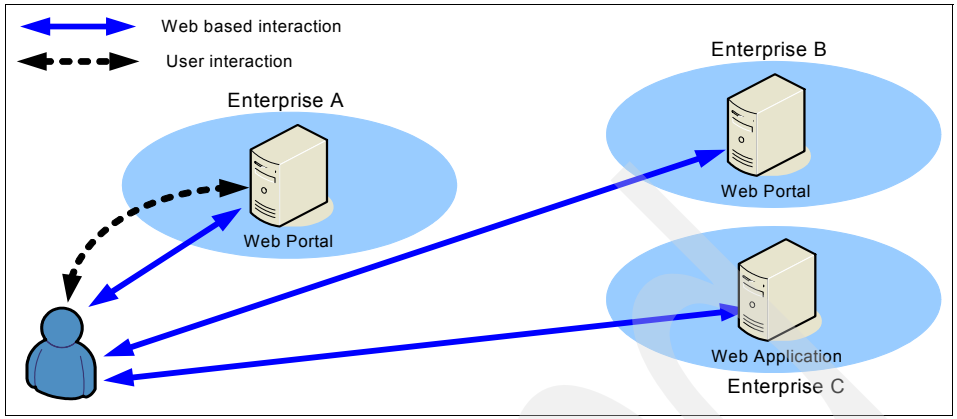


Figure 23-9 Secure user interaction - F-SSO

A simplified view of a user interaction is illustrated in Figure 23-9, where a user interacts with Enterprise A who acts as the IdP and two businesses Enterprise B and C who act as SP's. The user interactions are all Web browser based and F-SSO is used to reduce sign-on for the user. The reduced sign-on may be accomplished with any of the SSO protocols, SAML, Liberty ID-FF or WS-Federation, see Table 23-1 on page 704 for help on selecting SSO protocol suitable for the federation partnership to be set up.

In the attempt to explain the different functionality in Federated single sign-on, the example in 23.1, "Federation example" on page 680 will be used in this section.

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

23.4.1 Push and pull SSO

There are two different ways of doing SSO, push and pull. Pull SSO is available in SAML 1.x and 2.0, Liberty ID-FF and WS-Federation. Push is available in SAML 1.x (with custom coding in Liberty ID-FF) and WS-Federation, see 23.3, "FIM standards and efforts" on page 698 for details.

Push SSO means that the SSO exchange is triggered by a request to the identity provider, which then PUSHes a security token (or an artifact that can be used to obtain the Security Token) to the service provider.

Pull SSO means that the SSO exchange is triggered by a request to the service provider, which then PULL's a security token (or an artifact that can be used to obtain the Security Token) from the identity provider.

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

23.4.2 Account linking

When a user has multiple login accounts at various sites or companies, navigating between these Web sites can be a cumbersome activity, not to mention the poor user experience. The user has to remember multiple site identity account names and passwords to access services on these Web sites. Account linking provides a simple mechanism for the user to link these distinct identity accounts that they have with different Web sites as long as the various companies or Web sites agree to this concept. The purpose of account linking is to deliver a single sign-on user experience with these various providers who are part of this agreement. Once accounts are linked, the user can authenticate to one provider and then navigate seamlessly to various service providers with whom they have linked accounts without having to re-authenticate or enroll.

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

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

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

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

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

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

23.4.3 Where are you from?

Some service providers may have trust relationships with multiple identity providers. This means that a user may possibly initiate SSO from one of many IdP's. For the service provider, the process of determining which IdP to request SSO from is referred to as *Where are you from?* (WAYF). This is a process by which a user may specify a preference for a given IdP for SSO purposes. This information is maintained by the SP so that it can easily determine, without user interaction, which IdP to request SSO from for future requests.

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

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

In some cases, a service provider may not be willing to expose a list of trusted identity providers. In this case, John would be given instructions by RBBanking to

directly access his identity provider (RBTelco) and initiate a SSO request through an identity provider based mechanism.

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

23.4.4 Session management and access rights

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

This implies that the service provider is able to manage some level of attributes or credentials for a user. These attributes are used to determine a user's local access privileges. Access privileges may be asserted by the identity provider in the form of asserted attributes about a user, such as group membership. This information may be used by the service provider as an indication of the types of actions considered allowable by the identity provider (or, actions that will be honored by the IdP on the user's behalf). The service provider is able to honor or disregard these attributes as required for its local behavior.

23.4.5 Logout

In some federation scenarios, the notion of single logout (SLO) is also required, allowing a user to invoke a logout of all sessions asserted by a given identity provider. Global logout can be requested by a user from either the IdP or an SP; the process of global logout is controlled by the identity provider. The IdP is responsible for maintaining a list of all SPs to which the user under went SSO in a given session. The IdP will then send a logout request to each of these SPs on behalf of the user.

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

Global logout does not imply that local logout goes away. It is possible that a user will want to log out of a session at a service provider without destroying their session at the identity provider. Note that this requires that the user know and understand the nature and workings of the federation. The more likely alternative to a local logout at a service provider is to provide a shorter session

lifetime/inactivity time out than is used in a standard, directly authenticated session. A shorter inactivity time out for an SSO user may be acceptable, as the user is not forced to explicitly re-authenticate. Instead, the SP will simply re-request an SSO from the user's IdP.

23.4.6 Credentials clean up

Logout, be it global or local, often implies the destruction of a session at a service provider. This session is often maintained at the edge of a network and may be independent of sessions with back-end applications. Back-end application sessions may be used to maintain a state between request/responses of a multi-step transaction. Logout, at both the identity provider, and service provider should ensure that not only edge sessions, but back-end application sessions (and session tracking artifacts), are destroyed.

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

23.4.7 Global good-bye

Global good-bye deals with de-provisioning of a user's access rights and entitlements within a federation scenario. Global good-bye is used when a relationship between an identity provider and a service provider is broken, all of the user's attributes (including transactional, profile and provider specific attributes) that are relevant to the destroyed relationship are also destroyed. Note that federation relationships may be terminated in several ways: A user may chose to terminate his binding of an identity provider to a service provider or an IdP and SP may chose to no longer do business together, breaking the binding for all of the IdP's users.

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

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

23.4.8 Account delinking

Account delinking is the process by which the common unique identifier is destroyed, removing the ability of an IdP and SP to uniquely refer to a given user. One result of account de-linking is that a user will no longer experience SSO from the IdP to the SP. Note that account delinking is independent of how a user's account/registry record was created at the service provider, meaning that account delinking is possible whether an account was explicitly created by a user and then linked, or created based on provisioning from the IdP to the SP. After delinking an account, a user or service provider may choose to link an account with a different identity provider, or the SP may choose to resume direct authentication of the user.

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

23.5 Web services security management

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

Web services security management functionality allows the establishment and management of federation relationships for application to application interactions, see Figure 23-10 on page 712, enabling the required trust and security. In this solution, an application is able to generate a Web services request, acting as a Web services client. This request can then be secured (encrypted and signed) to provide message-level confidentiality and integrity.

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

Web services security management adds the ability for message-level authentication and authorization, in the context of a federation relationship. This is studied in detail, out of a IBM Tivoli Federated Identity Management product point of view, in 24.4, “Web services security management” on page 766.

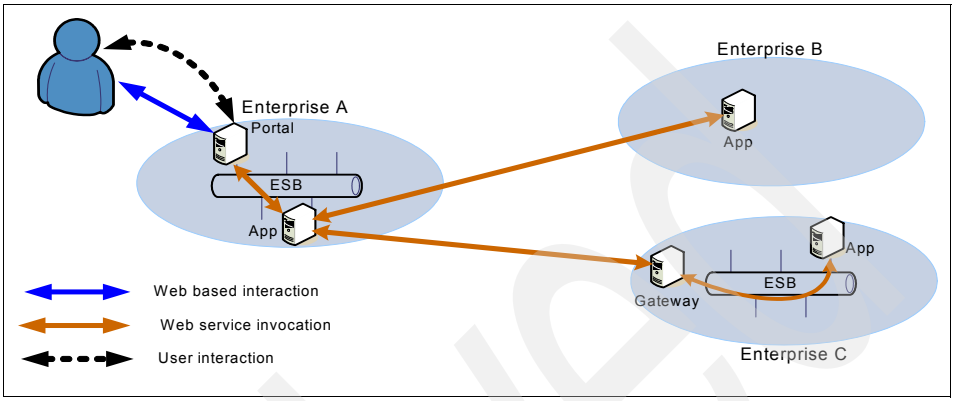


Figure 23-10 Secure business interaction - Federated Web services security

A simplified view of a user interaction is illustrated in Figure 23-10, where a user interacts with the portal in Enterprise A. The portal renders an application that uses Web services to integrate with any of the two businesses Enterprise B and C, which have exposed an application as a Web service. The interactions are all application-to-application based and Web services security management is used to enable security in the end-to-end integration.

In the attempt to explain the different functionality in Web services security management, the example in 23.1, “Federation example” on page 680 will be used in this section.

Technology relevant to Web services security management are as follows: Web services, WS-Security and gateways.

23.5.1 Web services

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

Web services technology enables businesses to describe available services and provide access to those services over standard Web protocols and communications boundaries. Web services has inherited and learned from the

way the World Wide Web revolutionized how people talk to systems. The new customers and business models, extensions of opportunity, new transparency and improved collaboration within enterprises and in some cases simplification in infrastructure and sometimes reduced cost. The key to these successes was a general server-to-client model in a highly distributed environment, and most importantly based on simple open standards and industry support.

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

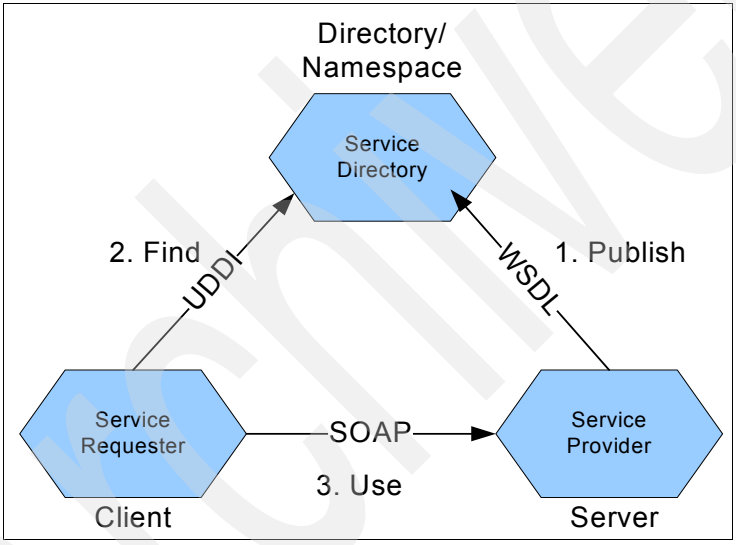


Figure 23-11 Basic Web services

Figure 23-11 shows the basic interaction model supported by Web services. Basic Web services define interactions among service requesters, service providers, and service directories as follows:

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

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

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

For more information about Web services, see the IBM Redbooks publication *Using Web Services for Business Integration*, SG24-6583, or Web services architecture - W3C Working Draft at the following Web address:

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

23.5.2 Web services security

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

For more on the SOAP message security specification is called “Web Services Security: SOAP Message Security 1.0”, and it can be found at:

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

This standard defines a set of SOAP extensions, seen in Figure 23-12 on page 715, that provide the ability to do the following:

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

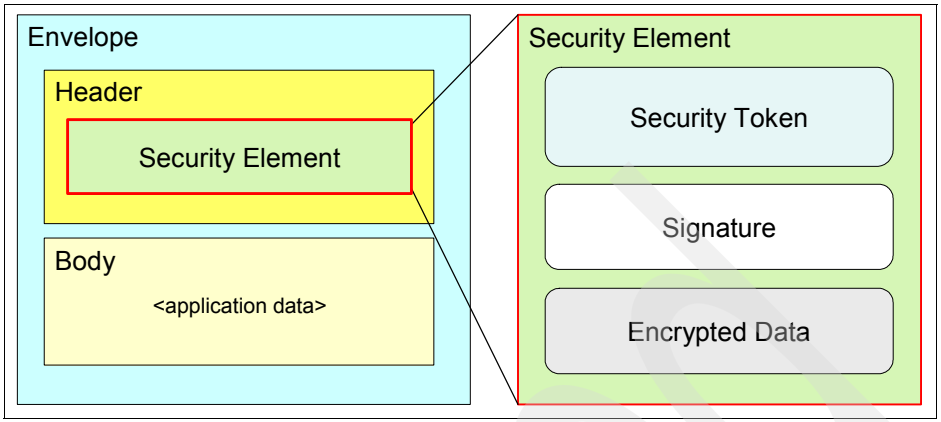


Figure 23-12 WS-Security: SOAP message security, extensions to the header

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

When RBTelco securely passes the client identity and attribute information to RBStocks, the request will use Web Services Security Management on the outbound side. A SAML assertion is added as a security token in the Web services request and then signing and encrypting it.

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

23.5.3 Web services gateways

For service providers, a Web services gateway (also known as XML gateway or XML firewall) acts much the same as an HTTP reverse proxy. Rather than proxying the HTTP protocol, the Web services gateway proxies SOAP traffic. SOAP (Simple Object Access Protocol) is a protocol for exchanging XML-based messages over computer networks and forms the messaging foundation for Web service communications.

A Web services gateway examines Web service requests leaving the boundary of an enterprise network or entering the boundary of an enterprise network. It

inspects the XML messages and performs access control checks based on policies configured in the authentication and authorization services.

RBTelco deployed a Web services gateway that it uses for inbound Web services requests from BigCorp. This gateway controls access to the RBTelco's Web service-based offerings as well as providing identity transformations to map the identity supplied by BigCorp in the request to a locally understood identity at RBTelco.

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

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

For more details on how to use the IBM WebSphere DataPower® XML Security Gateway XS40 as a Web services gateway in the context of federated identities, see 24.4.3, "Web services gateway" on page 771.

In section 26.2.3, "XML gateway pattern" on page 828, there is a more complete discussion of the service requester/provider scenarios and there are some examples of gateways available in the market.

23.6 Federated identity provisioning

Provisioning is about remotely having the capability of managing attributes of for example a user as part of an identity management process. The same provisioning definition is also valid for provisioning of other services or resources for example applications or servers. Within federated identity management the focus is on the user/identity. This is studied in more detail, out of a IBM Tivoli Federated Identity Management product point of view, in 24.5, "Provisioning services" on page 775.

In the attempt to explain the different functionality in provisioning, the example in 23.1, "Federation example" on page 680 will be used in this section.

Federated identity provisioning extends these provisioning management activities beyond an internal trust domain, see Figure 23-13 on page 718. Federated identity provisioning makes it possible to extend local account

provisioning at an identity provider to include federated account provisioning out to multiple service provider partners. A service provider, when notified of the federated provisioning request, can perform the local provisioning necessary to supply its service to the specified employee.

When used with provisioning of account data and authentication credentials, provisioning solutions generally come in one of two flavors: *Runtime* (or just-in-time) and *a priori provisioning*. Runtime provisioning solutions are also referred to as enrollment solutions as a user is registered, or enrolled, for a set of services, as part of the fulfillment of a single sign-on request. Sometimes this is referred to as *silent registration* because the users do not see a separate registration/enrollment step in their user experience.

A priori provisioning is the process by which a user account creation request can be sent to federation business partners outside of the scope of a single sign-on request. This allows both the identity provider and service provider to create local accounts/registry records for a user in response to some action at the IdP. A priori provisioning is often triggered by an account creation event at the identity provider. A priori provisioning may also be triggered by other events, such as a change in a user's status that in turn gives him access to more business partner resources, or a subscription event by a user, signing up for services that the identity provider in turn out sources to a third-party service provider. Note that like runtime provisioning, a common user identifier is established for a user automatically as part of a priori provisioning.

Runtime, or just-in-time provisioning allows a service provider to create a user account/record in her local registry in response to a single-sign-on request from a trusted identity provider. This may happen when an SP receives a SSO request from a trusted identity provider but does not have any record of the user claimed in the SSO request. Instead of rejecting the SSO request, the SP may choose to create a user record based on the claimed *common unique identifier* (CUID). The CUID-local identity mapping is therefore established at this time; in fact, the SP is not required to ever establish its own, non-CUID local identity for this user.

In the case of BigCorp, provisioning a new employee within the BigCorp system will cause account creation of the user's BigCorp required accounts. A federated provisioning solution could also cause the sending of a provisioning trigger request to RBTravel, but in this case just-in-time provisioning is used instead and the user is provisioned at RBTravel on the fly if no user exists there. As this account is created during the single sign-on from the user in BigCorp, the common user identifier information will have been included with the provisioning request and so no account linking step is required by this new user.

Provisioning solutions allow the identity provider to create or update a user's transactional attributes, such as entitlements to service providers, as required. These attributes are typically managed by the end user's identity provider. In the

case of BigCorp, employee Mr. Employee One may have a corporate credit card used for travel purposes. If this credit card number changes, BigCorp may be required to provision this transactional attribute to BigCorp's travel agency RBTravel. Similarly, Mr. One's salary may be considered a transactional attribute, as it will be used by benefits providers to determine Mr. One's eligibility for services. As such, it must be provisioned to BigCorp's benefits providers if/when it changes.

Provisioning requests sent between identity providers and service providers must be secure and be based on open standards. A standard that satisfies these requirements is WS-Provisioning. See Figure 23-13. These requirements may be satisfied by an implementation of the WS-Provisioning standard. WS-Provisioning is a specification authored by IBM to provide a Web service interface to communicate provisioning requests and responses. See 23.3.8, "WS-Provisioning" on page 703 for more details on the WS-Provisioning standard.

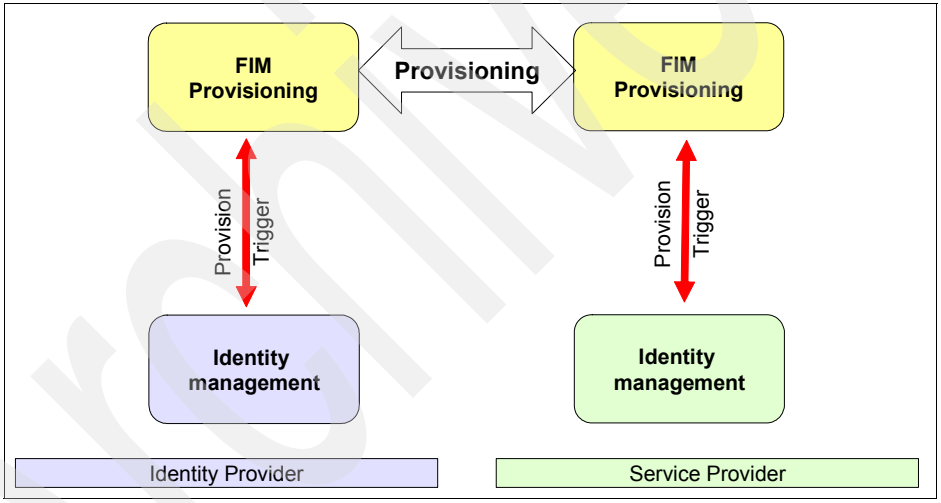


Figure 23-13 Federated provisioning overview

WS-Provisioning includes operations for adding, modifying, deleting, and querying provisioning data. It also specifies a notification interface for subscribing to provisioning events. Provisioning data is described using XML and other types of schema. This facilitates the translation of data between different provisioning systems.

23.7 Conclusion

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

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

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

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

24

Federated Identity Manager

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

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

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

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

24.1 Federated Identity Manager functionality

Tivoli Federated Identity Manager delivers a key functionality called Trust Service to enable identity federation solutions. This service is the basis for providing federated provisioning, Web single sign-on, and Web service security management solutions. Each of these solutions may be deployed independently or can be deployed together within a SOA environment to deliver standards-based identity federation solution.

As shown in Figure 24-1, Tivoli Federated Identity Manager provides overall functionality for identity federation that includes the following:

- Federated provisioning
- Web single sign-on (SSO)
- Web services security

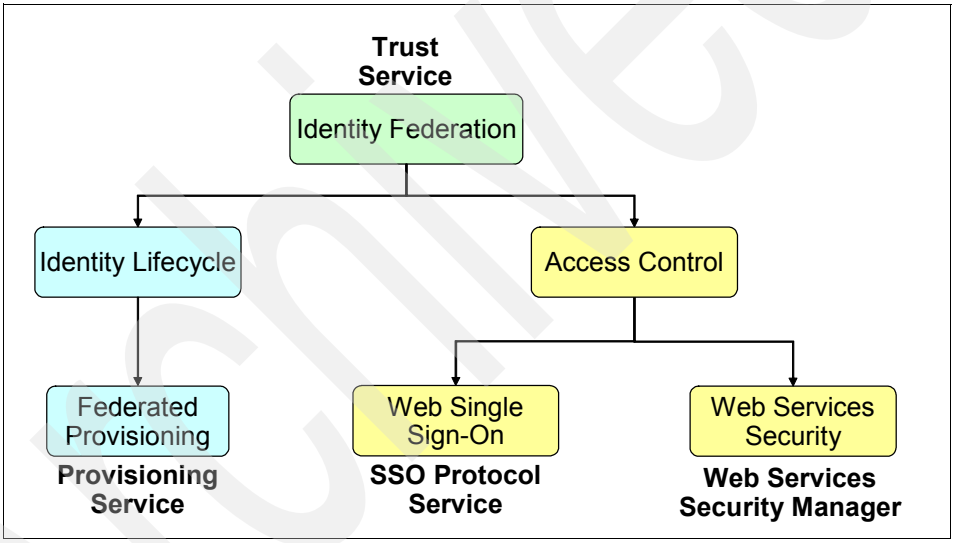


Figure 24-1 Tivoli Federated Identity Manager runtime services

Federated Identity Manager service components are described in 24.2, “Federation services” on page 723. These components represent individual services that may exist as distinct services or as logical services within Tivoli Federated Identity Manager. Each of these functional components is represented by a logical service, so that the following applies:

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

IBM also offers the Tivoli Federated Identity Manager Business Gateway, a suitable offering to deliver cross-company single sign-on for small-to-medium businesses or cross-domain single sign-on for department or project-level requirements. This offering commonly fits organizations looking to support standards-based federation protocols with a minimal footprint and without the requirements of a highly available deployment.

Another gateway that plays an integral part of some Web services security solutions is the WebSphere DataPower XML Security Gateway XS40. This hardware device provides Web services access control, XML encryption and digital signature, WS-Security, and content-based routing. The XS40 may call out to the trust service in Tivoli Federated Identity Manager to perform complex identity mappings, mediation, and authorization to access Web-applications and services in a SOA environment. The WebSphere DataPower XML Security Gateway XS40 can be used whenever an architecture calls for a Web services gateway.

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

24.2 Federation services

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

- Engage in trust relationships that facilitate direct integration of business processes in the most efficient fashion. The concept of business federations directly provides services for customers registered at other (business partner) businesses or institutions by establishing business trust relationships.
- Share identity information and entitlements in a trusted fashion between companies. Current approaches to identity management generally rely on companies incurring user lifecycle management costs by maintaining redundant identities to manage employees, business partners, and customers. The relationship between the business and these individuals can change fairly frequently. Each change requires an administrative action that can result in a high cost of user lifecycle management.
- Exchange, in a secure and trusted manner, tokens referring to a Principal, their attributes, privileges, and so on. These tokens are used to communicate

information used for the authentication and authorization of a Principal to a business partner.

- Maintain security in a Web services oriented architecture, allowing for secure standards based application-to-application inter-enterprise communication.

The following sections give an overview of each of the services components represented in Figure 24-2, which is the Federated Identity Manager services architecture used in Tivoli Federated Identity Manager. The complete set of Tivoli Federated Identity Manager services allows for creation of federated SSO, Web services security management and provisioning solutions. The dark-grey boxes are non core Tivoli Federated Identity Manager services that are used as part of different Federated Identity Manager solutions.

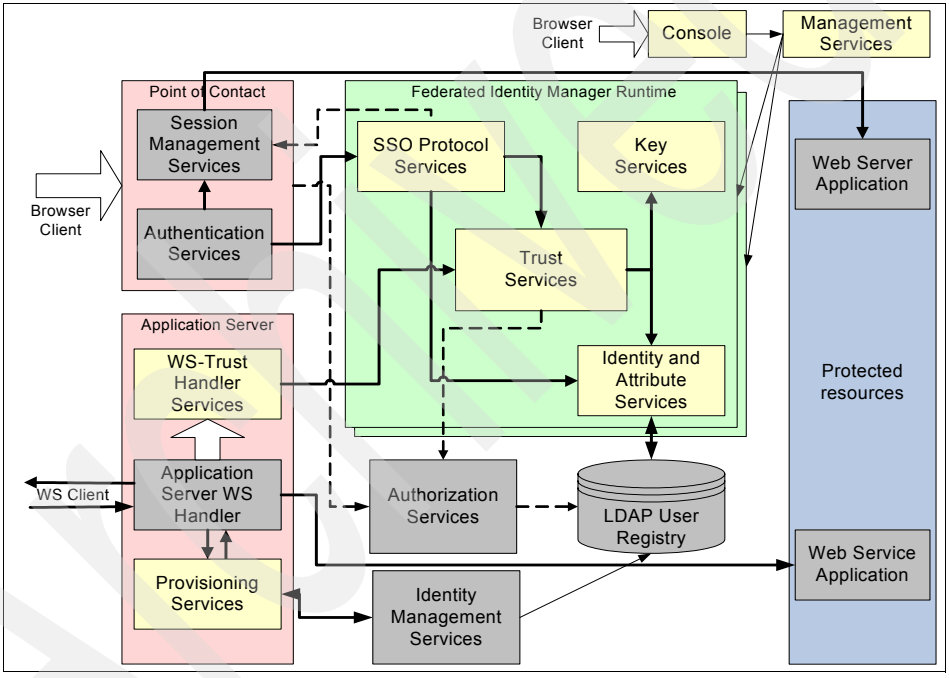


Figure 24-2 Federated Identity Manager services architecture—The full picture

A different view of the services is found in Figure 24-3 on page 725, where the layers Point of Contact (PoC), SSO protocol service (SPS) and Trust service are shown in their external communication interfaces over standardized protocols. Both user and application based interactions are shown, since they differ in layering and protocols.

In the application-based interaction to the left in Figure 24-3 on page 725 the PoC is represented by a Web services (WS) handler interfacing with the Trust

service, and may be represented by a WS gateway. The provisioning service may be viewed as an application exposed as a Web service.

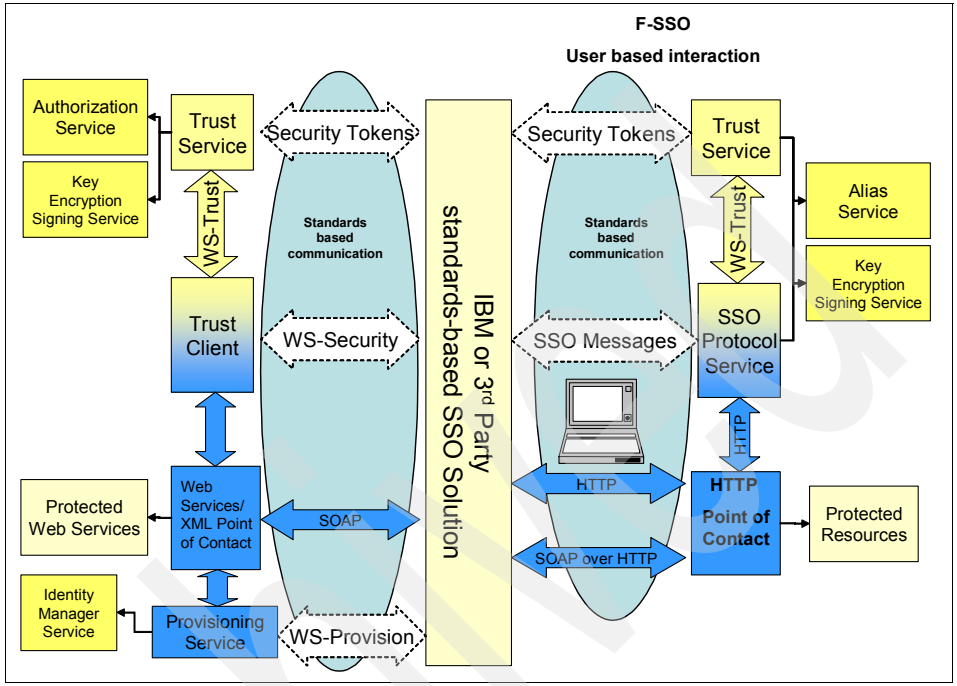


Figure 24-3 User and application based interaction components and their communication

24.2.1 HTTP point of contact

The *HTTP point of contact* is used for HTTP-based user interactions. The point of contact service provides authentication service and the session management service functionality. These services are typically provided by Tivoli Access Manager for e-business through the Access Manager for e-business reverse proxy or the Access Manager for e-business Web plug-in. With the recent introduction of the Federated Identity Manager Business Gateway offering, the point of contact service may be Microsoft's Internet Information Service (IIS) or the IBM WebSphere Application Server, thereby, removing the dependency on Tivoli Access Manager for e-business.

Authentication services

Authentication services provide the functionality required to evaluate and validate user-provided credentials. Authentication services evaluate credentials such as a username and password, secure ID token pass phrases, X.509 certifications, Kerberos ticket and so on, provided by the *user agent* on behalf of a user.

Authentication services are able to invoke some back end data store such as a LDAP registry, or a secure ID token server, to validate these credentials.

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

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

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

Instead of incorporating support for each of these federation protocols (both the interaction with the business partner and the evaluation of the presented token) within the authentication service, an external SSO service is used. SSO services (described in 24.2.3, "Single sign-on protocol services" on page 728) provide the run time for the federation protocols necessary to implement the challenge/response interaction with a third party.

In response to the evaluation of user provided or federation provided authentication credentials, an authentication service will generate information that is used by a session management service to govern a user's session. This information is typically represented as a set of user credentials, or user privileges. This information is used by a session management service and by *authorization services*, as described in section 24.2.7, "Authorization services" on page 735.

Session management services

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

Sessions are created at a Session Management Service in response to a successful authentication event. These events may include the initial user authentication, step-up authentication, re-authentication or a successful security token validation amongst others. Implementations of Session Management Services often incorporate *authentication services*, so that an authentication service exists as a logical service; however, this is not always the case.

24.2.2 SOAP/XML point of contact

Web services gateways are used for federated application-to-application communications and serve the role of an *XML point of contact*. A Web services Gateway is much the same as an HTTP reverse proxy. Figure 24-4 shows the similarity between the HTTP reverse proxy and the Web services gateway as a point of contact.

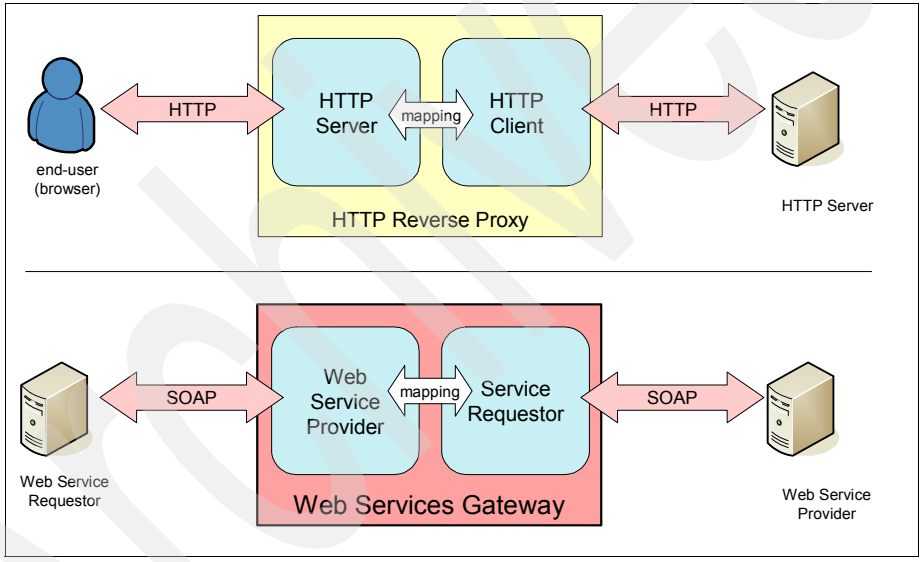


Figure 24-4 Web services Gateway: A reverse-proxy for Web services

Among the key features, the XML point of contact service provides filtering of bad requests, Web services access control, XML encryption and digital signature, WS-Security, and content-based routing. These services are typically provided by the IBM WebSphere DataPower XML Security Gateway XS40.

While the WSSM solution is part of Tivoli Federated Identity Manager and discussed elsewhere in this document, the WebSphere DataPower XML Security Gateway XS40 is a separate hardware solution that tightly integrates with Tivoli Federated Identity Manager. For more information about these products see 24.4.3, “Web services gateway” on page 771.

Web services gateways offer the following deployment characteristics:

- Decouple deployment from invocation

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

  - Process abstraction

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

  - Flexibility

    As a service provider, you need the flexibility of changing your deployment infrastructure without notifying all the service requestors. Say a Web service is deployed in a machine that later fails during operation. There needs to be a process to route the invocations to an alternate service in your infrastructure.

- Protocol transformation

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

24.2.3 Single sign-on protocol services

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

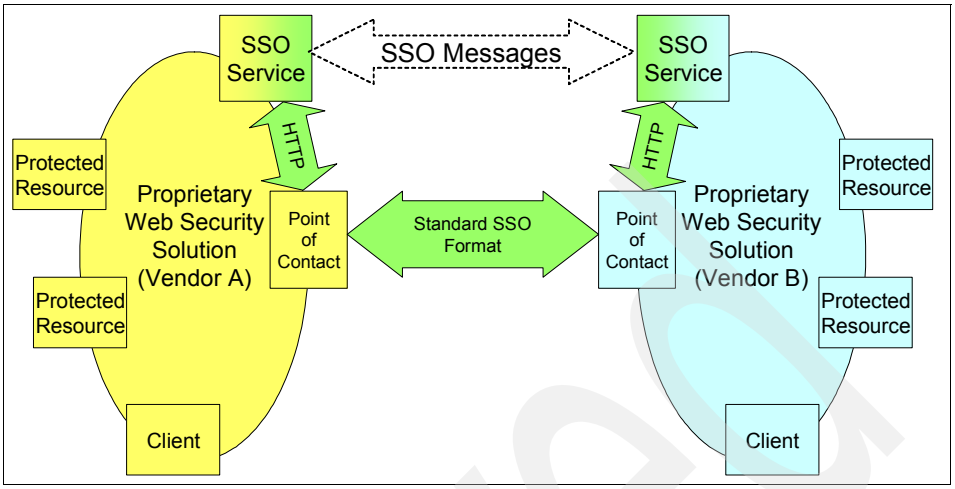


Figure 24-5 Externalized SSO services

In Tivoli Federated Identity Manager, the responsibility for handling SSO protocol messages is off-loaded from the point of contact server as shown in Figure 24-5. SSO protocol endpoints are instead hosted by a separate service, the SPS. The point of contact server still maintains control of user sessions, providing session management services.

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

External Authentication Interface

Tivoli Federated Identity Manager provides an authentication mechanism through its SPS with the capability that allows clients to sign in with credentials generated by another party—the identity provider. By integrating Tivoli Federated Identity Manager with the point of contact, the F-SSO can be treated as just another point of contact authentication mechanism, thus having the SPS create a point of contact login session. When used with Tivoli Access Manager as the point of contact service, the External Authentication Interface (EAI) is used as the integration point with Tivoli Federated Identity Manager. See 9.4.6, “External Authentication Interface” on page 297 for a detailed description of the EAI.

24.2.4 Trust services

Federation relationships require a trust relationship-based federation between business partners. A trust relationship is represented by the combination of the security tokens used to exchange information about a user, the cryptographic

information used to protect these security tokens (and the communications used to broker token exchange) and optionally the identity mapping rules applied to the information contained within this token.

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

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

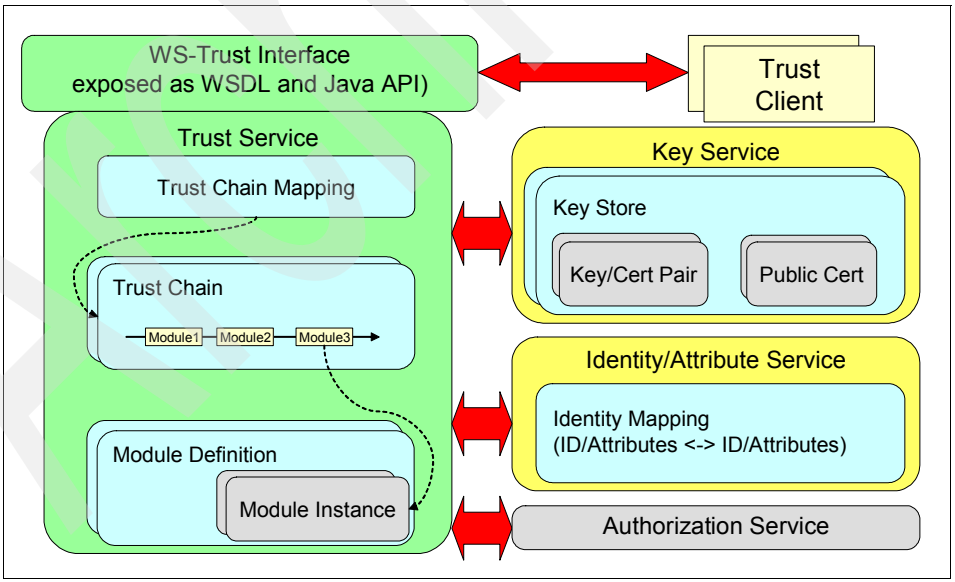


Figure 24-6 Trust service components and connections

Figure 24-6 on page 730 shows the logical components and connections of the Tivoli Federated Identity Manager trust service. The trust services performs security token related function, such as token creation, validation and exchange, also it does authorization for Web services. The trust service is accessed by trust clients using either SOAP requests or direct JAVA API calls.

Trust service modules

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

Trust service modules can make calls out to other Tivoli Federated Identity Manager components. For example, most token modules call the *key service for signature creation and validation*. Liberty token modules call out to the identity service for alias lookup. Access Manager credential modules and authorization modules call out to the authorization service. Support is provided for the following token modules within Federated Identity Manager 6.1:

- SAML 1.0, 1.1 and 2.0 (Generate and Consume)
- Liberty 1.1 and Liberty 1.2 Assertion (Generate and Consume)
- Access Manager Credential (Generate and Consume)
- JAAS Subject (Consume)
- Kerberos Token (Consume)
- X.509 Token (Consume)
- RACF PassTicket (Generate and Consume)

Support for custom modules (for additional token support, customization of mapping functions, and custom trust chains) is also provided.

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

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

The Tivoli Federated Identity Manager trust service has a very flexible identity mapping function that allows it to exchange tokens using a different identity

mapping rule with each partner. The trust service mapping module is called to perform the mapping and it looks up the configured identity mapping for the partner in question.

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

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

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

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

Figure 24-7 on page 733 shows how the trust service performs a token exchange. Trust chains like the one shown here are used for all federated SSO operations. These trust chains are created automatically when you configure federated SSO.

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

The STSUUSER document is now used as input to the identity mapping module. The mapping used by the module is particular to the partner we are dealing with

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

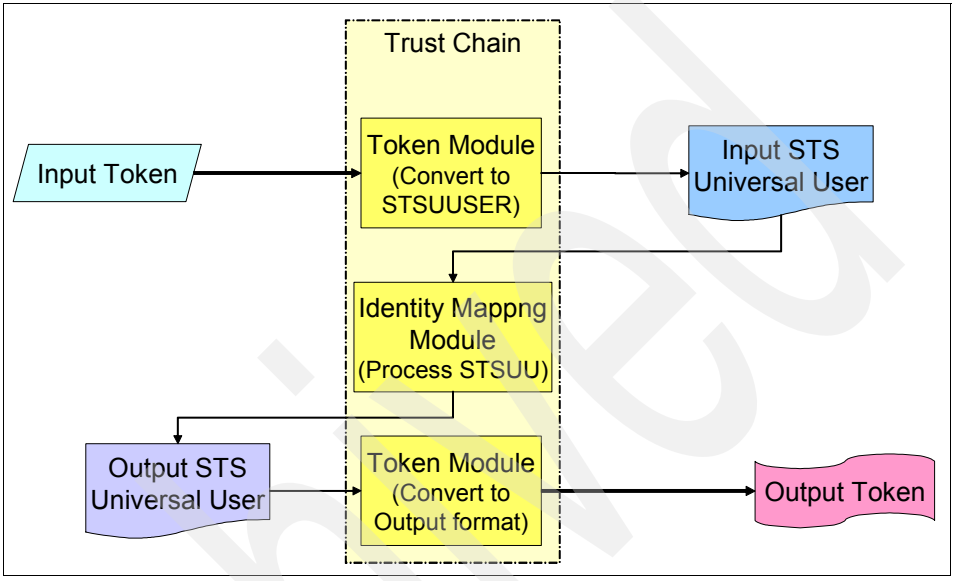


Figure 24-7 Trust service processing for F-SSO

Figure 24-8 on page 734 shows how the identity mapping module is implemented using an XSL parser.

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

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

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

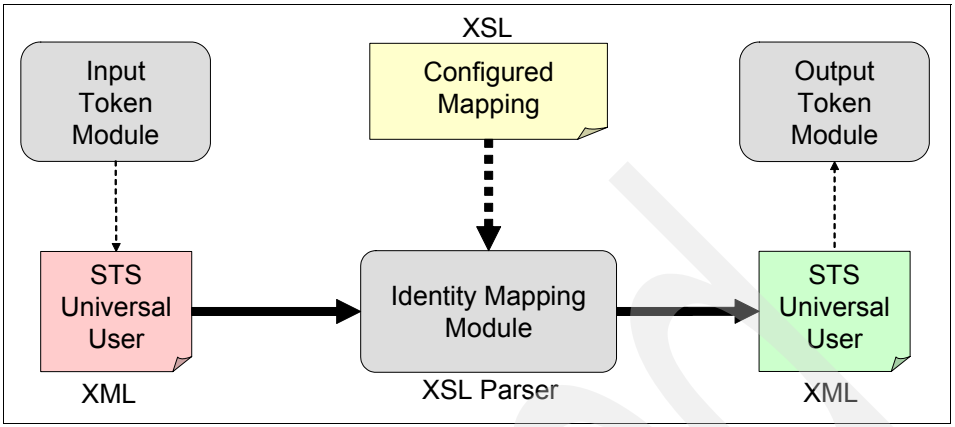


Figure 24-8 Trust service transformation engine

24.2.5 Key services (KESS)

Key services are leveraged to provide access to key stores used by a trust service and the SPS. This allows the trust service and SPS to plug in/access different key stores as required. It also provides a single point through which key management may be accomplished. Key services are often implemented as logical components within a trust service.

24.2.6 Identity services

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

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

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

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

Alias services

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

24.2.7 Authorization services

Authorization services are responsible for providing access decision point functionality within a security model. The authorization service itself may not act as an *access enforcement function* (AEF). AEF functionality is typically provided by Session Management Services. Tivoli Access Manager provides AEF functionality with Tivoli Access Manager WebSEAL acting as an access decision point (ADP).

At their simplest, authorization services implement an access decision functionality, taking in a request for access and evaluating this request based on a user's session privileges. The authorization service may respond with a simple yes/no, indicating if an access request is allowed or not. Based on this response, session management services act as the authorization enforcement point by allowing/disallowing the actual request for access.

24.2.8 Provisioning services

Provisioning services are used within a federated environment for both a priori and run-time provisioning solutions. Provisioning services interact with both local identity management systems (such as Tivoli Identity Manager, see Chapter 17, “Identity management” on page 509, and Chapter 18, “Identity Manager structure and components” on page 547, for more information about Tivoli Identity Manager) and local data stores (access via identity services). Provisioning services are leveraged to federate local identity management systems across federation business partners and to provide federated management of identity data, including transactional and profile attributes.

Provisioning services are leveraged as part of the identity management functionality within an enterprise; as such, they are often integrated with a local identity management system. This allows a local identity management system to

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

24.2.9 Management services

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

- ISC: The new IBM Integrated Solutions Console providing a single portal style administrative console for Tivoli Federated Identity Manager
- API: Used by for example the InfoService, see 24.3.6, "InfoService" on page 763

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

Console

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

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

After the ISC is installed, console plug-ins are deployed into the ISC, see Figure 24-9 on page 737. The Federated Identity Manager Console is one such plug-in. The ISC is accessed over HTTP(S). This means that the Tivoli Federated Identity Manager administration console can be accessed from any client that has connectivity to the machine where the ISC is installed.

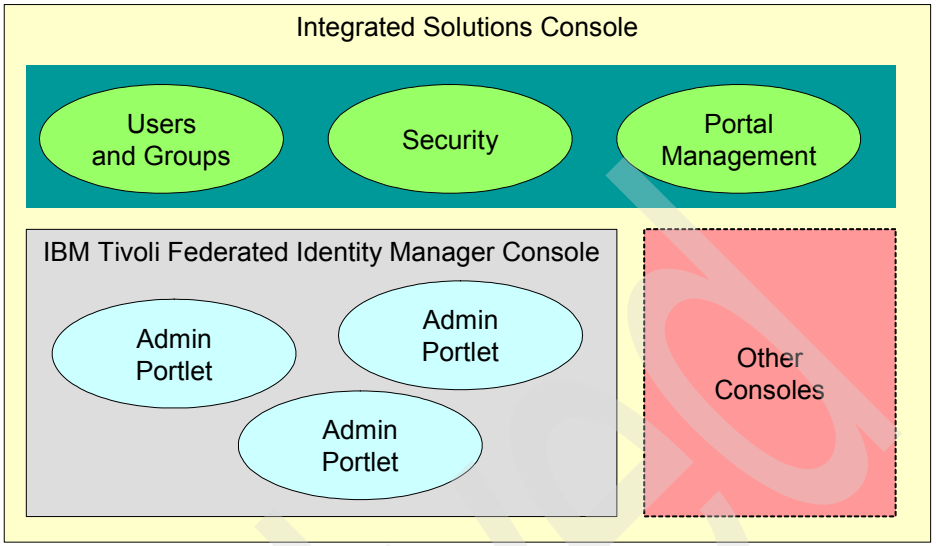


Figure 24-9 IBM Tivoli Federated Identity Manager Console within the ISC

Deployment manager

The ISC console interface uses the *deployment manager* to push deployment and configuration to remote Tivoli Federated Identity Manager nodes, as shown in Figure 24-10 on page 738. The deployment manager supports multiple domains and clustered nodes, more on clustered nodes in 26.1.5, “Highly available architecture patterns” on page 815. WebSphere Application Server functionality used to synchronize the configuration files to clusters and the Tivoli Federated Identity Manager runtimes on the WebSphere Application Servers read the files locally.

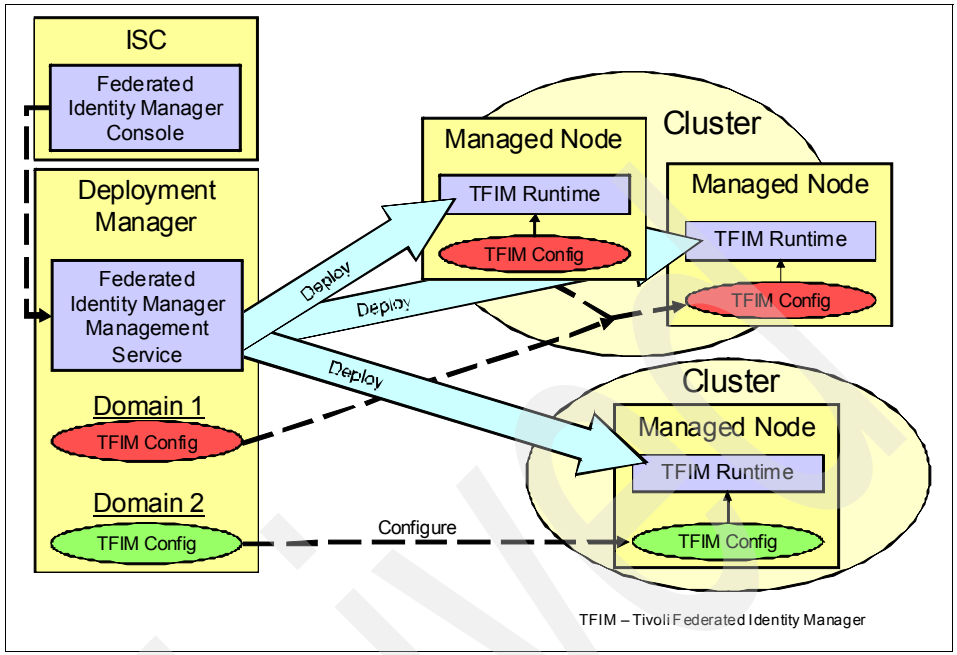


Figure 24-10 Federated Identity Manager deployment and configuration

24.2.10 Audit Services

Tivoli Federated Identity Manager (unlike the Federated Identity Manager Business Gateway at this time) can use the Common Auditing and Reporting Service for consolidating and centralizing audit log information.

Common Auditing and Reporting Service

The Common Auditing and Reporting Service has the following features:

- Provides auditing support
  - Defines a consistent format for events that can be audited using the Common Base Event (CBE) format
  - Provides a centralized collection point for events that can be audited from various sources
  - Provides consistent management of the lifecycle of audit data
- Facilitates reporting of audit data
  - Provides interfaces to stage audit data into custom report tables
  - Enables customers to use a reporting tool of their choice to build custom audit reports

  - Facilitates cross-product audit reports
  - Exploits IBM products to provide audit reports for immediate use
- Provides interfaces for IBM products to create and submit data that needs to be audited

For more information about the Common Auditing and Reporting System, refer to Chapter 27, “Introducing IBM Tivoli Common Auditing and Reporting Service” on page 845.

Federated Identity Manager provides support for sending audit events to the local file system or to the central Common Auditing and Reporting Event Server. The following section outlines the events that can be audited from within Federated Identity Manager.

Auditable Events

Federated Identity Manager contains various points of interest from an auditing perspective. When used in conjunction with the Web security server audit records, some analyses needs to be performed about the optimal point for configuring auditing. A good example is the complexities around authentication.

- As a service provider, where single sign-on to the point of contact server is provided through the federated single sign-on protocols, it is necessary to audit the events at the Federated Identity Manager product rather than the point of contact. Auditing the event at the Web security server point of contact only provides a credential inheritance event rather than an authentication event.
- As the identity provider, the authentication event actually occurs at the Web security server, so appropriate Common Auditing and Reporting audit needs to be configured there rather than the federated single sign-on service. At the single sign-on service perspective, it simply receives a single sign-on token from the Web security server, which is not considered part of the authentication event.

By using the Common Auditing and Reporting solution, a customer has the opportunity to combine audit records from Web security servers, such as WebSEAL, with Federated Identity Manager records to produce more meaningful audit reports for management.

That being said, it is worth considering the audit points that are available with Federated Identity Manager and what events are available at those points so that the audit solution can be architected appropriately.

Audit events that can be configured to be generated from Federated Identity Manager include:

- Single sign-on audit event: This audit event shows the authentication information from a resulted SSO operation. It includes information about partners, the initiating Web security server, and so on.
- Single logout audit event: This audit event includes all information related to a user initiated single logout event.
- Name Identifier Management audit event: This audit event can be generated when a user identity mapping is created, when a user consents to federation, when a de-federate operation occurs, and when a user mapping is updated. Obviously, when an audit event of this type is generated, the event contains tags that identify the operation as well as all other information relevant to the type of operation that occurred.
- Trust service events: These audit events are generated when a Federated Identity Manager server validates a token, issues a token, maps an identity or authorizes a Web service access.
- Encryption audit events: These audit events are generated whenever encryption events occur.
- Signing audit events: These audit events are generated whenever data is signed.
- Management audit events: These audit events are generated when a Federated Identity Manager server creates a new federation, modifies an existing federation, a federation is deleted, a partner is added or deleted, when the properties of a partner are modified, and when Web service partners are created or deleted.

It is up to the implementor to customize their own audit reports for Federated Identity Manager. Instructions for doing so are outlined within the User Guides.

24.3 Federated single sign-on

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

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

When considering an SSO solution, there are two main areas where participants must agree on the technology choice in order to achieve interoperability.

The first area is the format and content of the security token that will be passed between the partners. The security token generated by the sending partner must be understandable by the receiving partner. Also, there must be an agreement as to what information is sent in the token and how it is interpreted. Typically the security token format is bound to the SSO protocol (SAML protocols use SAML assertions, Liberty ID-FF protocols use Liberty specializations of SAML assertions). With Tivoli Federated Identity Manager, security token generation and consumption is handled by the trust service as invoked internally by the SPS. This is discussed in more detail in 24.2.4, “Trust services” on page 729.

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

Note that an SSO standard does not only deal with a profile for SSO, but also profiles for single logout, federation and alias management. The SPS is also responsible for handling these messages. These areas are discussed more later in this chapter.

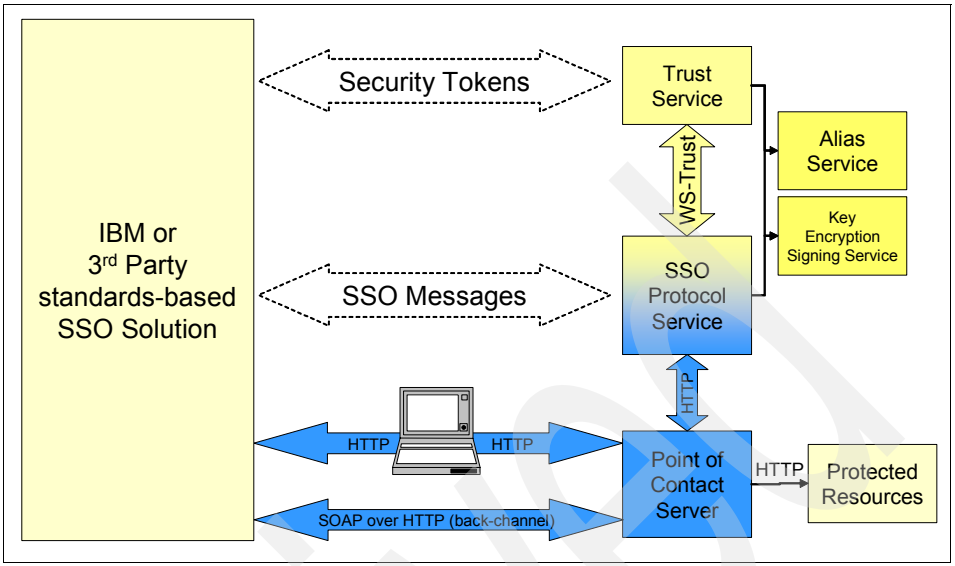


Figure 24-11 SSO components and communication

Figure 24-11 shows the communications and exchanges that take place at each layer of Federated Identity Manager when performing Web-based SSO. Note that no internal details are shown for the third-party side because their architecture is not known (and not important).

At the *communication* layer, HTTP messages are being handled by the point of contact server. In the Federated Identity Manager solution, this is either WebSEAL or the Web Server Plugins. For Federated Identity Manager Business Gateway, this can be either WebSphere Application Server or IIS (as a service provider only). All real communication is via the point of contact server. It must support the HTTP standard in order to interoperate with the client and with the third-party solution.

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

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

24.3.1 Architecture overview

Figure 24-12 shows the Federated Identity Manager architecture required to support Web-based SSO protocols such as Liberty, WS-Federation, and SAML 1.0, 1.1 and 2.0.

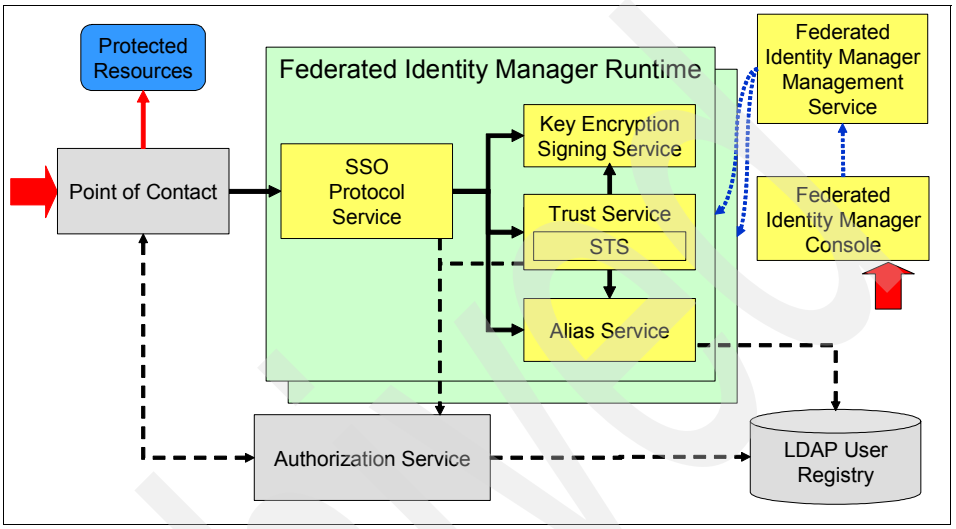


Figure 24-12 Federated Identity Manager components for federated SSO

All outside communication with the environment comes via the HTTP point of contact server. The following sections outline the configuration and message flows for a Federated Identity Manager configuration and a Federated Identity Manager Business Gateway configuration.

Federated Identity Manager message flow

Within Federated Identity Manager, the point of contact is WebSEAL or the Web Server Plugins for the remainder of this section, which uses WebSEAL in the description. WebSEAL maintains the Web session with the client and manages authorization. WebSEAL also triggers authentication (either local or SSO) when protected resources are requested. WebSEAL authorization is managed by the authorization service, in our case Tivoli Access Manager.

WebSEAL has a junction to the SPS. Incoming SSO messages are directed to the junction that connects to the SPS. WebSEAL will simply forward these as usual. WebSEAL can also re-direct the client to the SPS in order to initiate SSO processes itself.

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

in the authorization service to terminate WebSEAL sessions during single logout (SLO) operations. In latter versions of the SPS, the EAI function is used to logout users as an alternative to the Access Manager administration API.

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

Federated Identity Manager Business Gateway message flow

Within Federated Identity Manager Business Gateway, the point of contact is either WebSphere Application Server or IIS. Let us consider the two separately.

WebSphere Application Server

Within Federated Identity Manager Business Gateway, the WebSphere Application server can act as either the identity provider or service provider within a configured federation. As such, its features include the ability to provide authentication services as well as authorization, session management, and content presentation.

When acting as an identity provider, WebSphere Application Server must provide authentication services on behalf of the service provider partner. The actual authentication can be performed using any of the WebSphere Application Server out-of-the-box authentication solutions, as of WebSphere 6.1, this now includes SPNEGO support. After a user is authenticated, the request for F-SSO is sent to the SPS for token generation based on the partner configuration.

When acting as a Service Provider, WebSphere Application Server must provide unauthenticated access to the F-SSO endpoint. This enables the SPS to authenticate the incoming token and assert an identity to the point of contact server, in this case WebSphere Application Server. WebSphere Application Server is then responsible for managing the user's session, performing authorization and providing the requested content.

IIS

Within Federated Identity Manager Business Gateway, IIS is supported for use as a point of contact in a service provider configured federation. As such, IIS is not responsible for providing authentication services direct to the user, rather it must be able to communicate with the SPS to process incoming authentication tokens. Federated Identity Manager Business Gateway configured with IIS can be configured as a partner of either a Federated Identity Manager identity provider, a Federated Identity Manager Business Gateway identity provider, or a third-party identity provider.

As part of the Federated Identity Manager Business Gateway solution, a Web server plugin is provided to route configured F-SSO requests to the SPS. The SPS is configured to return headers that contain the required information for IIS to establish a session for a local identity. It then forwards the request to the appropriate target for content delivery.

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

24.3.2 Trust in F-SSO

Security tokens are included in a message to pass an identity and to convey security-specific information (used for authentication or authorization purposes, for example) about a requestor. See Figure 24-13. These tokens are common to (at least) one other business partner and contain pre-arranged security-relevant information.

These tokens are themselves protected through signing and encryption, often using the same keying material as used at the message layer. This information is part of the trust infrastructure in the same way that keys are used for signing/encryption purposes: The proper use of these tokens conveys information about the holder of these tokens. The trust service provides a means of managing these security tokens and the trust relationships bound to these security tokens.

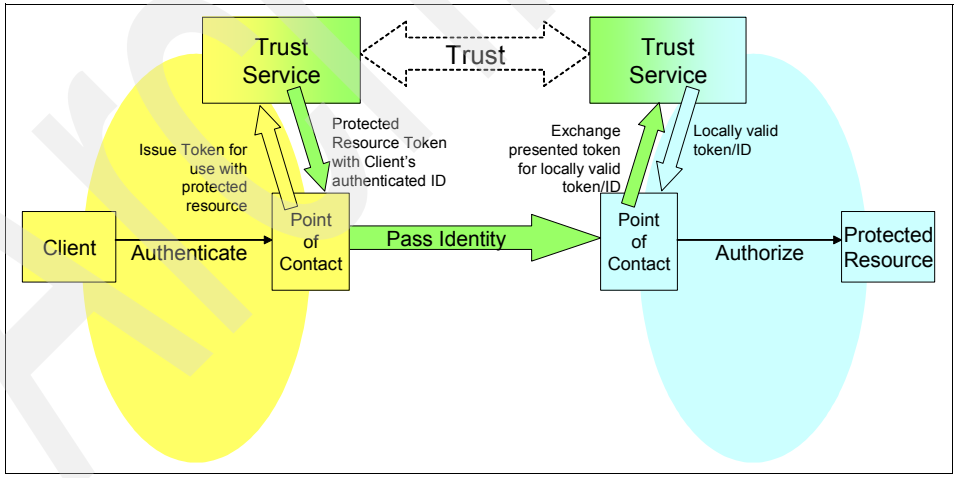


Figure 24-13 Using trust service in F-SSO

Token management is based on information such as the issuer of a token, the intended destination of the token, and the intended use of the token. This allows

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

24.3.3 F-SSO protocol functionality

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

Tivoli Federated Identity Manager Business Gateway provides a refined set of protocol support, without the requirement for Access Manager for e-business. The concepts presented within this section are equally applicable to Tivoli Federated Identity Manager as Tivoli Federated Identity Manager Business Gateway.

Single sign-on

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

In general, F-SSO protocols (as all other CD-SSO protocols) come in two flavors: *push* and *pull*.

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

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

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

Once a user's SSO information has been established and validated at a service provider, the service provider will maintain a local session (including credentials) for the user. This will allow the service provider to implement local access control policies, for example, for the user's session.

Single logout

Previous attempts at SSO have often neglected the corresponding single logout functionality. Logout can be of two forms: local and global. In general, logout from the user's identity provider should force a global logout, whether the user requests a global or local logout. This is a strong recommendation/requirement

that stems in large part from the liability normally assumed by an identity provider for a user within a F-SSO relationship.

It is not always the case that logout should be an allowable service provider action. This follows in that if a user has single signed-on to the service provider, he may well have no notion that he has a separate session with this service provider. Rather than confuse the user by offering a logout action at the service provider, we expect that most scenarios will set a short session lifetime (inactivity time-out) at a service provider and rely on SSO to re-establish a session at a service provider, perhaps many times within the lifetime of the user's identity provider session.

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

Note that logout in general has implications for things such as session duration (differing durations at identity providers and service providers). In general, the inactivity time-out set for an identity provider should be longer than that set for its service provider business partners. This will prevent a user from timing out at the identity provider when executing a lengthy transaction with a given service provider.

Account linking

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

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

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

Note that account linking is sometimes referred to as provisioning, where the linkage between existing accounts is the information being provisioned. This is *not* provisioning for two important reasons: One, it requires that the user already has pre-existing accounts at both the identity and service provider. Two, the account linking requires that a user be actively involved in the process of establishing the account linking at both providers.

Tivoli Federated Identity Manager does provide a Web services provisioning solution, as described in 24.5, “Provisioning services” on page 775. This Web services-based provisioning allows the linking of two identity management systems for a complete user lifecycle management solution, including the provisioning of information (attributes, subscriptions, account status, and so on) between federation business partners.

Password synchronization

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

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

We expect that password synchronization solutions will not be common. What is more likely is that a service provider will disable the password at the service provider side once account linkage has been accomplished (without disabling the user's account). This means that the user can only access the service provider

resources from their identity provider. If/when account de-linking (see the next section) occurs, user self-care can be invoked to allow the user to re-establish a password for local access.

Account de-linking (name de-federation)

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

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

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

Account de-linking may be triggered at the identity provider (for scenarios where the user is changing service providers or simply wants to remove F-SSO functionality between the IdP and SP) or at the service provider (when the user wants to establish a new IdP or wants to remove F-SSO functionality at that SP).

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

Where are you from

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

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

A more likely form of storing WAYF information is in the form of a domain cookie that identifies the user's identity provider and nothing else. There is no security-relevant information of any form stored in this cookie. If a user attempts to access a service provider resource and is not carrying some form of F-SSO token, the service provider will look for a WAYF cookie to determine the user's

home domain. Based on the identity provider information stored in this cookie, the service provider will be able to determine (based on local configuration) the corresponding F-SSO endpoint at the identity provider.

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

24.3.4 Point of contacts for SSO

As discussed, Tivoli caters to differing customer requirements by providing two F-SSO products, Tivoli Federated Identity Manager and Tivoli Federated Identity Manager Business Gateway. This section provides an overview of one of the fundamental differences between the two products, being the point of contact support each provides. While these offerings are separate products, the F-SSO concepts previously described are relevant in each implementation.

Tivoli Federated Identity Manager provides the run-time implementation of supported SSO profiles. Access Manager for e-business provides the HTTP point of contact functionality for Federated Identity Manager. As such, Tivoli Federated Identity Manager has dependencies on Access Manager for e-business, and Access Manager for e-business has dependencies on Tivoli Federated Identity Manager.

Tivoli Federated Identity Manager Business Gateway uses the WebSphere Application Server or Microsoft Internet Information Services as the point of contact server, while providing the suite of F-SSO protocol support through the Federated Identity Manager runtime. As such, there is no dependency on Tivoli Access Manager for providing the point of contact, authorization, or auditing capabilities. This is completely off-loaded as a responsibility of the point of contact implementation.

In the next section, we briefly discuss the relationships introduced in this section.

SSO with Access Manager for e-business

Federated Identity Manager relies on the point of contact for session management for all users, whether Federated Identity Manager is acting as the identity provider or service provider. As part of the user's session management, Access Manager for e-business is responsible for only allowing authorized users

to participate in SSO relationships (for example, not all of an identity provider's users may be entitled to F-SSO functionality) and will subsequently audit these accesses.

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

When configured in a service provider environment, Federated Identity Manager must be able to determine a user's local identity and create an Access Manager for e-business credential for this user. This implies that the SPS is an unprotected resource.

SSO with Microsoft Internet Information Services (IIS)

Federated Identity Manager Business Gateway can be configured with IIS as the point of contact server at the service provider. In this configuration, IIS is responsible for providing the authorization, session management and content delivery to the end user. When SSO requests are made to the F-SSO endpoint on IIS at the service provider, the request is routed to the Federated Identity Manager services for validation. As such, this resource is required to be unprotected.

SSO with WebSphere Application Server

Federated Identity Manager Business Gateway can be configured with WebSphere Application Server (stand-alone, no cluster support) as the point of contact server within a federation. This includes configuration support for the WebSphere Application Server to act as either the point of contact as an identity provider or service provider, while Federated Identity Manager Business Gateway provides the F-SSO services.

When configured as an identity provider, Federated Identity Manager Business Gateway expects that WebSphere Application Server will correctly authenticate users based on the customer's configuration requirements, and will assert this user's identity to Federated Identity Manager Business Gateway as part of an SSO request. Federated Identity Manager Business Gateway will provide the F-SSO token generation (and management) capabilities for the requested service provider partner. This implies that the Federated Identity Manager Business Gateway SSO endpoint be protected.

When configured as a service provider, Federated Identity Manager Business Gateway attempts to determine a user's local identity from the incoming token and create a local identity for the WebSphere Application Server. The

WebSphere Application Server is responsible for authorization, content delivery, and session management based on the established user identity. This implies that the Federated Identity Manager Business Gateway SSO endpoint be unprotected.

24.3.5 Federated single sign-on approaches

F-SSO may use a variety of methods to communicate and assert identity. The different methods will not have support for all functionality described in 24.3.3, “F-SSO protocol functionality” on page 746. The standards were introduced in 23.3, “FIM standards and efforts” on page 698, and some of the characteristics of each protocol are highlighted in Table 23-1 on page 704. For detailed examples, refer to Part 2 “Customer environment” in the IBM Redbooks publication *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394. In general, aside from proprietary solutions, there are three approaches to Web-based browser SSO and federation:

- SAML
- Liberty ID-FF
- WS-Federation

SAML

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

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

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

- Browser/POST profile
- Browser/Artifact profile

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

With a Browser/POST profile, a SAML assertion (vouch-for-token) is included in the response that is sent to the service provider as part of an HTML form as

shown in Figure 24-14 on page 754. This is called a *front channel* exchange of the SAML assertion.

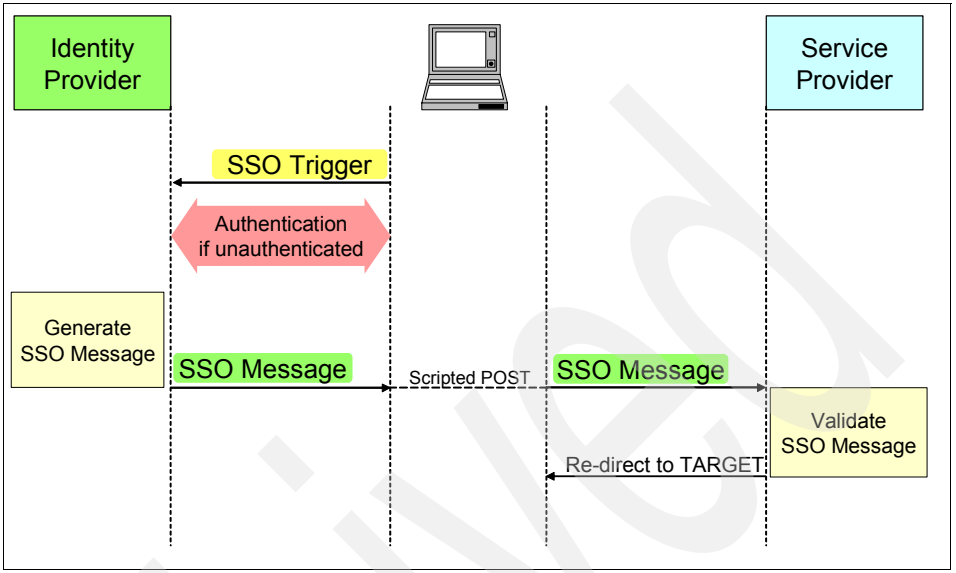


Figure 24-14 SAML SSO: Browser POST

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

Both SAML profiles are invoked by a user being directed to an *Inter-Site Transfer Service* at the identity provider. The Inter-Site Transfer Service is a URL that corresponds to a Federated Identity Manager endpoint.

In Figure 24-15, the Browser/Artifact profile is shown. The step wherein a direct SOAP/HTTP request is made from the service provider to the identity provider to exchange the browser-artifact for the appropriate SAML assertion is done over the mutually authenticated connection—the back channel.

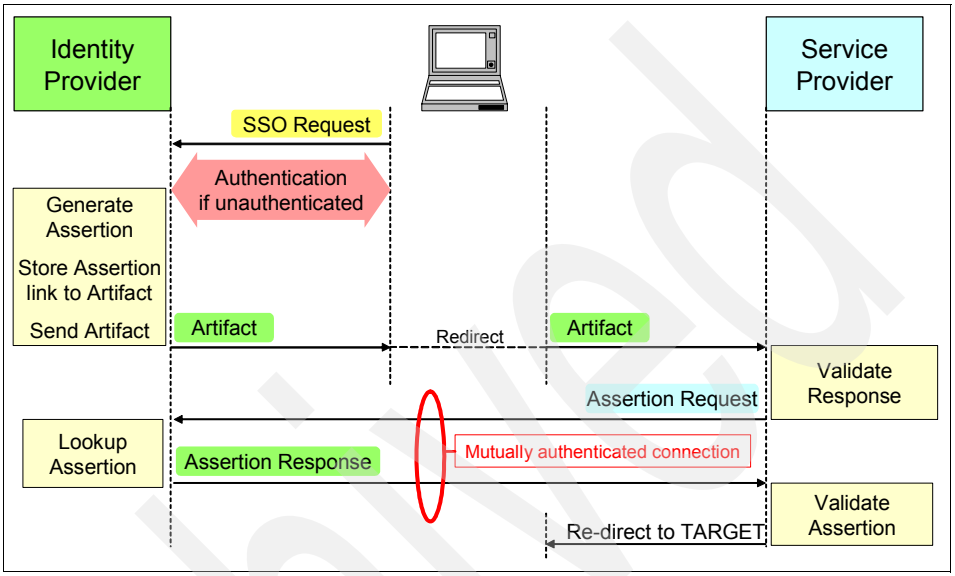


Figure 24-15 SAML SSO: Browser/Artifact

Liberty Alliance Identity Federation Framework

The Liberty Alliance Identity Federation Framework (ID-FF) extends SAML functionality beyond the push-based SSO of SAML. Federated Identity Manager SPS supports Liberty 1.1 and 1.2 ID-FF. The Federated Identity Manager trust service supports Liberty Assertions. The ID-FF defines:

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

The ID-FF SSO protocols have three flavors:

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

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

http://www.projectliberty.org/

Browser/Artifact SSO profile

The flows of the Liberty Browser/POST SSO profile are shown in Figure 24-16.

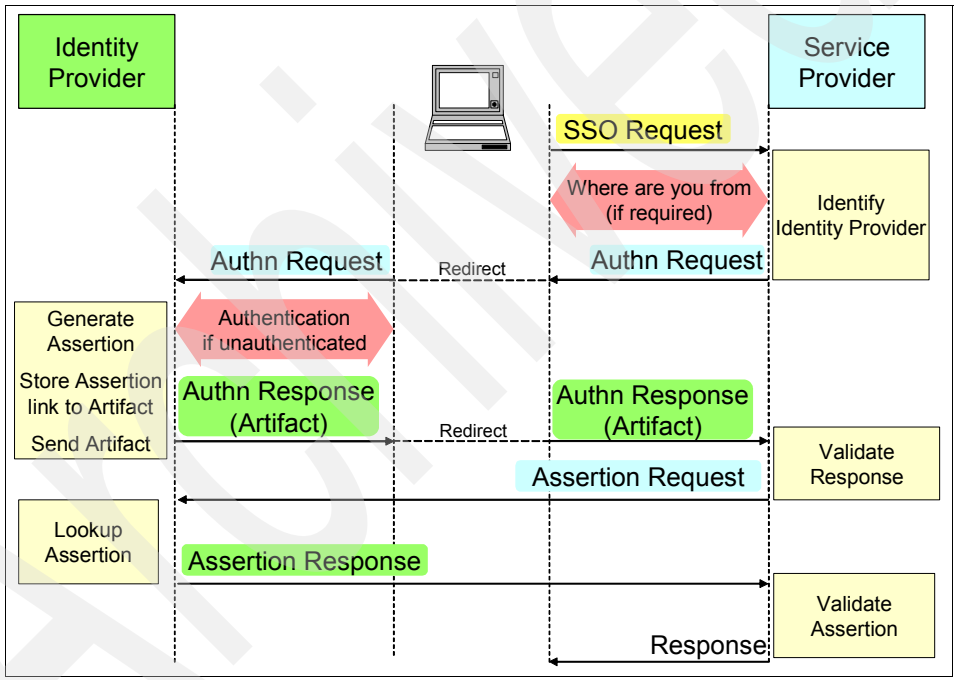


Figure 24-16 Liberty: Browser POST profile

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

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

require interaction with the user to prompt them for the appropriate identity provider.

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

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

Liberty Register Name Identifier

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

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

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

Liberty Federation Termination Notification

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

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

Liberty Single Sign-Out

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

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

Identity provider introduction

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

Liberty-enabled client/proxy

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

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

Figure 24-17 shows the role of Federated Identity Manager in a LECP profile, where a WAP Gateway is acting as the LECP. Note that in this scenario, Federated Identity Manager need only accommodate steps 4 and 6 when acting as an identity provider, and steps 1, 3, 7, and 11 when acting as a service provider.

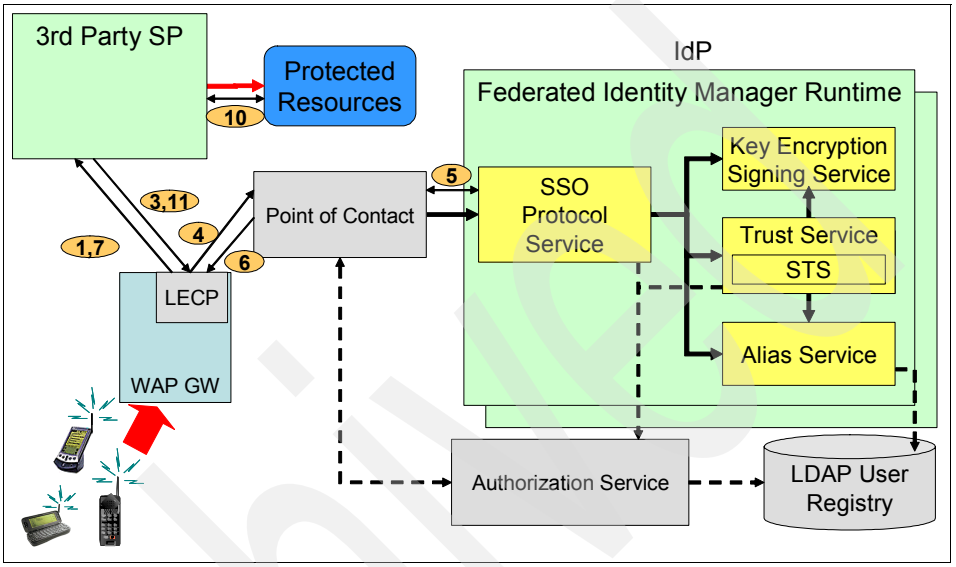


Figure 24-17 Liberty enabled client proxy (LECP) example

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

Unsolicited authentication response

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

WS-Federation passive client

The WS-Federation passive client specification, published by IBM and Microsoft, is available at the following Web site:

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

The specification states the following:

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

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

- Pull means that the SSO is initiated at service provider, the service provider determines identity provider then the service provider requests SSO from identity provider and identity provider responds with SSO token. See Figure 24-18.
- Push means that the SSO is initiated at identity provider and then the identity provider sends SSO token to service provider. See Figure 24-19 on page 761.

Pull

In Figure 24-18 an SSO is triggered at the service provider by sending a special SSO trigger message to the service provider WS-Federation endpoint. If the service provider has multiple identity providers configured then it must determine which to send the client to for authentication. It can do this either by reading a cookie set on a previous visit, checking for a parameter in the query string of the SSO trigger, or by sending the user a list of identity providers to choose from.

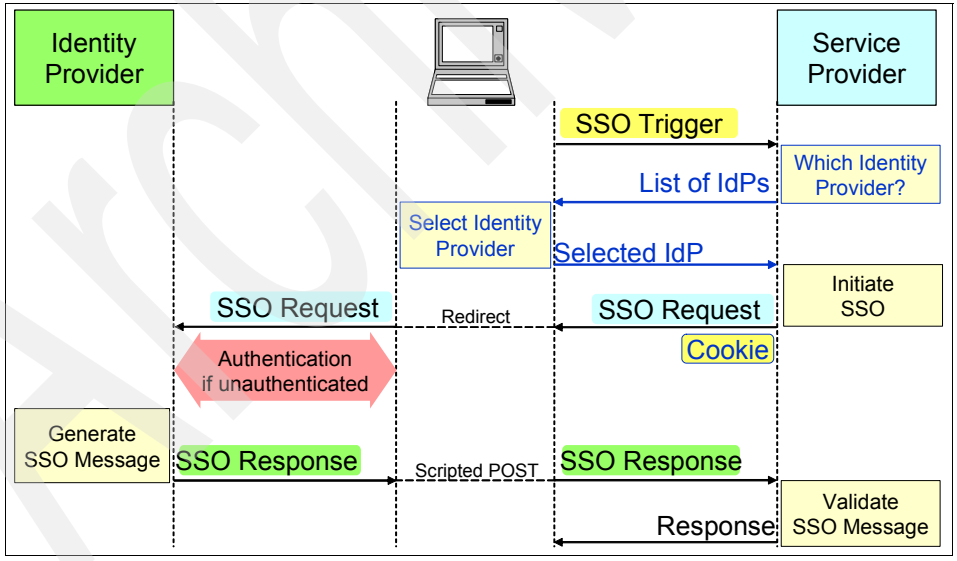


Figure 24-18 WS-Federation: Select ID Provider and SSO (Pull)

Once the service provider has determined the correct identity provider, it builds an SSO Request message which is send to the identity provider. The SSO message is send in the query-string of a re-direct to the WS-Federation endpoint

of the identity provider. A cookie set in the redirect identifies the identity provider. It is a persistent cookie which will allow the service provider to determine the correct identity provider next time without having to prompt the user. The SSO request shown here is being sent as a result of a redirect from the service provider.

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

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

Push

Figure 24-19 shows the protocol flow for a WS-Federation PULL operation. The WS-Federation protocol really starts with the SSO request received from the client. However, it is useful to see what causes the SSO request to be received, so this is also included.

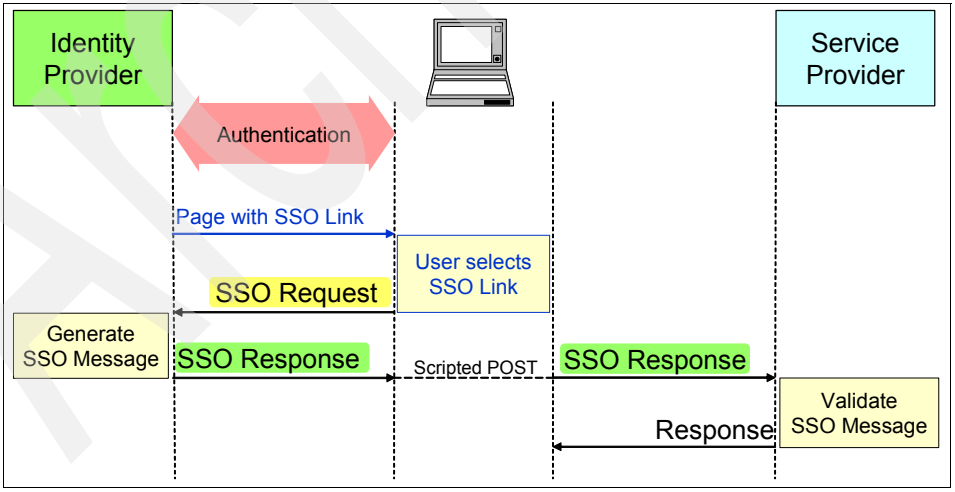


Figure 24-19 WS-Federation: SSO (Push)

It is unlikely that a user would manually type an SSO request message into their browser (although they could). It is much more likely that an identity provider will include a link on their site that a user can select in order to access some service provider resource (for example, for BigCorp you would see - *Click here to book a hotel with our preferred partner RBTravel*). Rather than direct the user straight to the service provider (only for it to have to direct the user back to perform SSO), this *special* link generates an SSO request to the WS-Federation endpoint of the identity provider, which immediately triggers the SSO exchange.

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

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

SAML 2.0

SAML 2.0 is a merge of SAML 1.0 and 1.1 with the specifications provided by Liberty 1.x. The emergence of SAML 2.0 is shown in Figure 24-20.

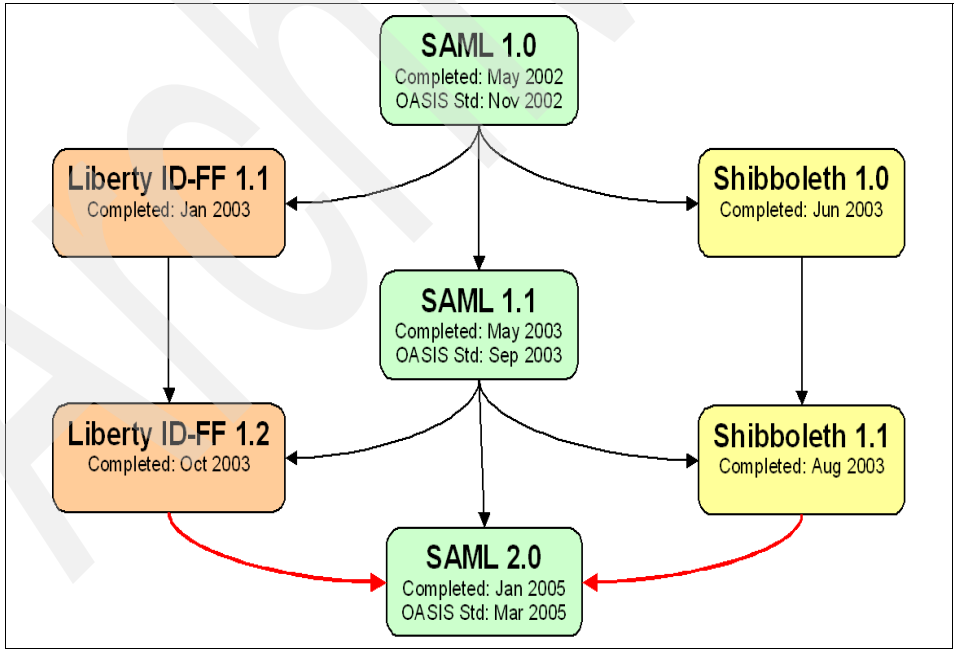


Figure 24-20 Standards evolution towards SAML 2.0

SAML 2.0 is a progression of these standards. The following is a brief description of the profiles supported by SAML 2.0. They include the following:

- Web browser SSO profile. This includes the browser artifact and browser post protocols as defined in “SAML” on page 753. Also added within SAML 2.0 is the ability to perform service provider initiated SSO as well as the associated message techniques (including re-direct, POST and artifact).
- Enhanced client or proxy profile: This profile defines the mechanism by which mobile devices, with limited user agent functionality, can operate in a federation.
- Identity provider discovery profile: This profile defines the “Where are you from?” message formats, allowing for the identification of your preferred identity provider site for single sign-on.
- Single logout profile: This profile defines messages for performing session logout on all services where a single sign-on operation occurred.
- Name identifier management profile: This is the account management profile used for account linking, alias management, delinking, and so on.

SAML 2.0 also provides the following additional features:

- XML format for storing away federation relationships. This makes it simpler for customers to manage their federations and reduces user error, allowing for definitions of federation attributes to be distributed to partners in an XML format.
- Provides the ability for encryption.
- Provides support for consent to federate. This removes the ability of a service provider to force a user to federate without the user’s knowledge. If enabled within the protocol, the IdP prompts the user to federate.
- Forced authentication: Even if a user is authenticated at the IdP, the SP can order the IdP to authenticate the user again.
- Passive IdP: The SP can force the IdP to not authenticate a user upon a redirect. This assumes that the user is already authenticated, and thus is required to return a valid token.

As you can see, SAML 2.0 merges the sets of profiles found most appropriate by the field, with some additions. Naturally then, it is emerging as the protocol of choice for customers.

24.3.6 InfoService

The InfoService is used to build a user interface reflecting the users defined federations. If a portal has many services where user have the possibility to use

F-SSO then it is necessary to be able to present the choices in a relevant manor, as not to confuse the user.

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

The InfoService makes Web services calls to the Management service to get this information. See Figure 24-21.

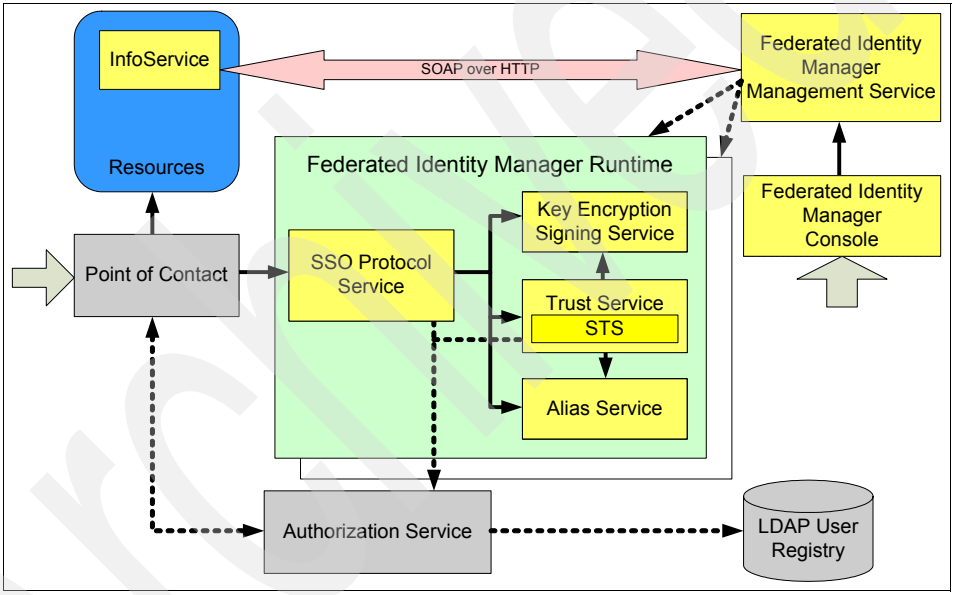


Figure 24-21 InfoService access to the Management Service

24.3.7 Specified level view of F-SSO architecture

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

The specified view for a Federated Identity Manager architecture for F-SSO is shown below in Figure 24-22 on page 765. A specified view describes the key nodes and the connections between them.

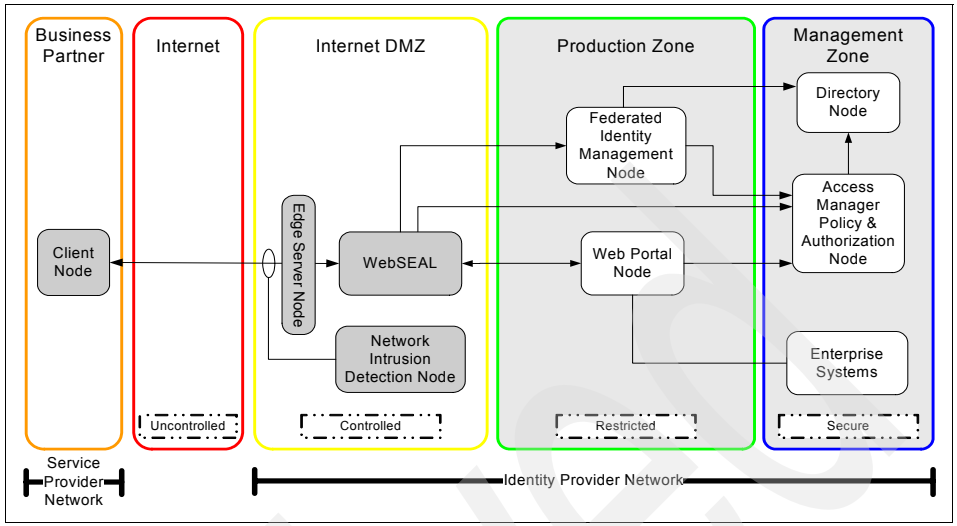


Figure 24-22 Generic Federated Identity Manager specified level view of F-SSO

The specified view for a Federated Identity Manager Business Gateway architecture for F-SSO is shown below in Figure 24-23. Care must be taken when deploying the Business Gateway solution in an HA environment. Support is only provided in a stand-alone server configuration, so when considering the protocols to use for SSO, if there is a dependency on any cluster function (such as in SAML browser-artifact exchange) this may have implications for the design.

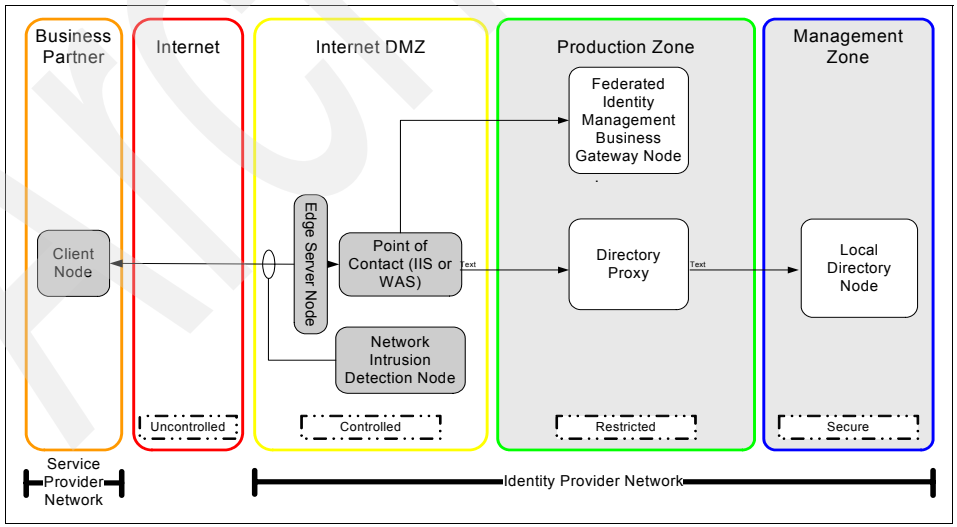


Figure 24-23 Federated Identity Manager Business Gateway level view of F-SSO

A more detailed look at F-SSO deployment is available in section 26.1, “Federated SSO architecture patterns” on page 804.

24.4 Web services security management

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

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

Figure 24-24 on page 767 shows the communications and exchanges that take place at each layer of Tivoli Federated Identity Manager when performing Web services Security Management.

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

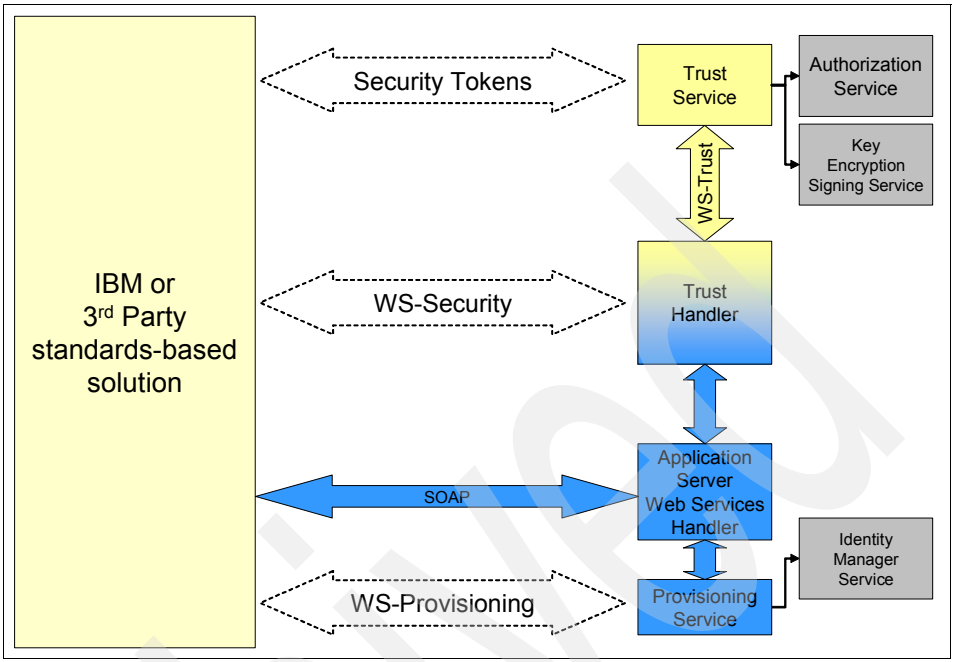


Figure 24-24 Web services security: Components and communication

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

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

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

24.4.1 Architecture overview

Figure 24-25 shows the components required for Web services security management with Tivoli Federated Identity Manager.

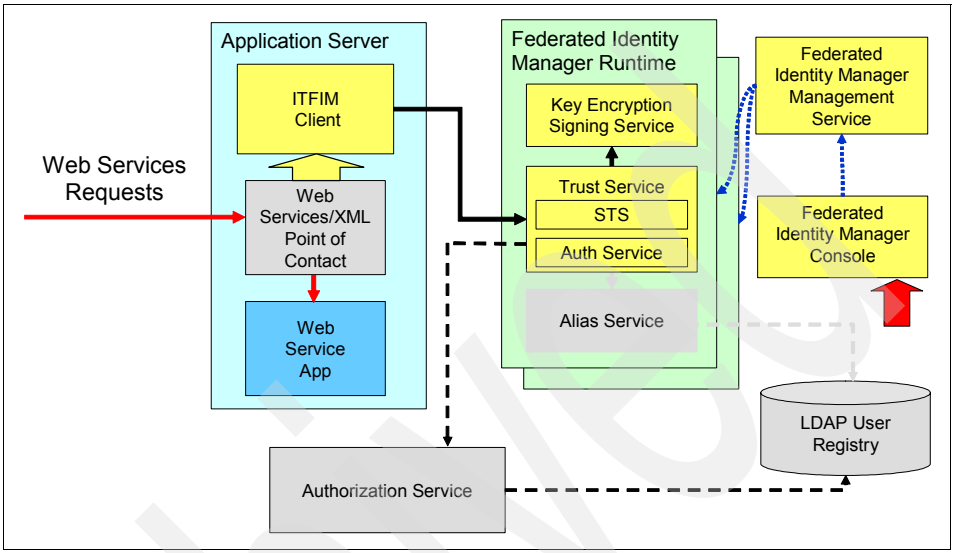


Figure 24-25 Components for Web services security management

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

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

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

Figure 24-26 on page 769 shows a user at Company A, accessing a resource at Company B via a Web service request.

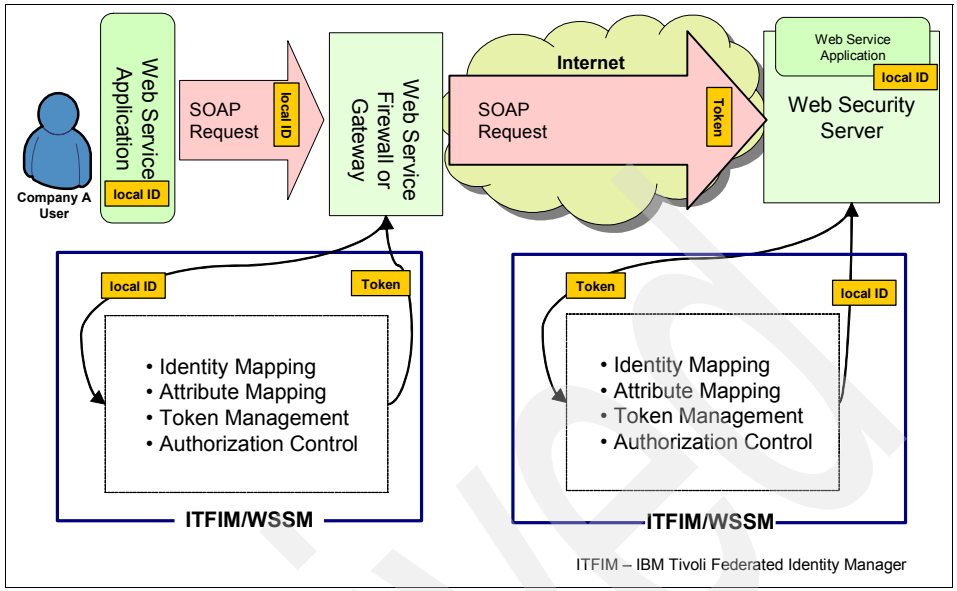


Figure 24-26 Web service security management (WSSM): Solution Architecture

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

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

24.4.2 WS-Security

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

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

In Figure 24-27, end-to-end message security is illustrated. The lock on the SOAP message is meant to imply that the SOAP message is inherently secure in and of itself. The SOAP message can be transported in any way and its security is not affected. The SOAP message could be sent as an e-mail attachment, carried on a floppy-disk, and so on, and the properties of privacy, integrity, proof of origin are not affected.

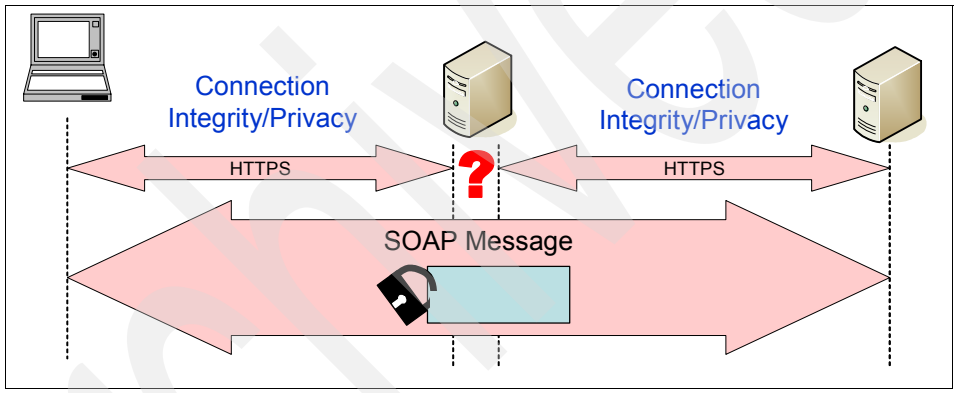


Figure 24-27 Message-based Security: End-to-End Security

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

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

For more on the topic WS-Security, and SOAP header extensions, see 23.5.2, “Web services security” on page 714.

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

24.4.3 Web services gateway

As mentioned in 24.2.2, “SOAP/XML point of contact” on page 727, the Web services gateway acts as a reverse proxy for SOAP traffic for service providers. On the Web services requestor side, an XML gateway can be used as an outgoing proxy for Web services. The primary IBM product used as a Web services gateway is the WebSphere DataPower XML Security Gateway XS40.

In the context of federated Web services, the purpose of this device is to do the following:

- Filter out bad requests
- Provide identity mapping between security domains
- Execute authorization decisions on Web service requests

As a hardware device, the XS40 is optimized for XML processing. This makes it ideal as a high-volume point of contact, which can rapidly filter out bad requests. Even though it offers its own identity mapping and authorization functions, a robust enterprise solution is to use the XS40 as the XML point of contact and configure it to call out to Tivoli Federated Identity Manager for identity mapping, mediation, and authorization.

Take the case of a request coming in from the external service provider. As shown in Figure 24-28 on page 772, 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 username 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*.

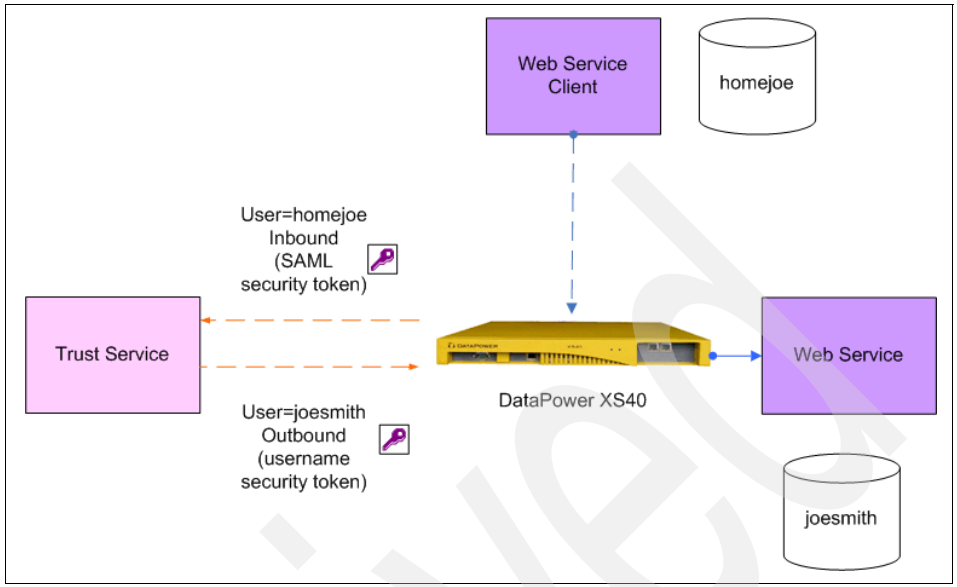


Figure 24-28 DataPower XS-40 as Web services gateway

In addition, an authorization service may be employed to verify whether the requesting entity is allowed access to the requested service. The trust service queries the authorization service with the name of the requester and the resource being requested. The authorization service returns a yes/no response and the trust service communicates the results back to the Web services gateway. In this way the gateway prevents unnecessary Web service requests from burdening the XML point of contact service with extra load. Figure 24-29 on page 773 extends the previous example to include authorization.

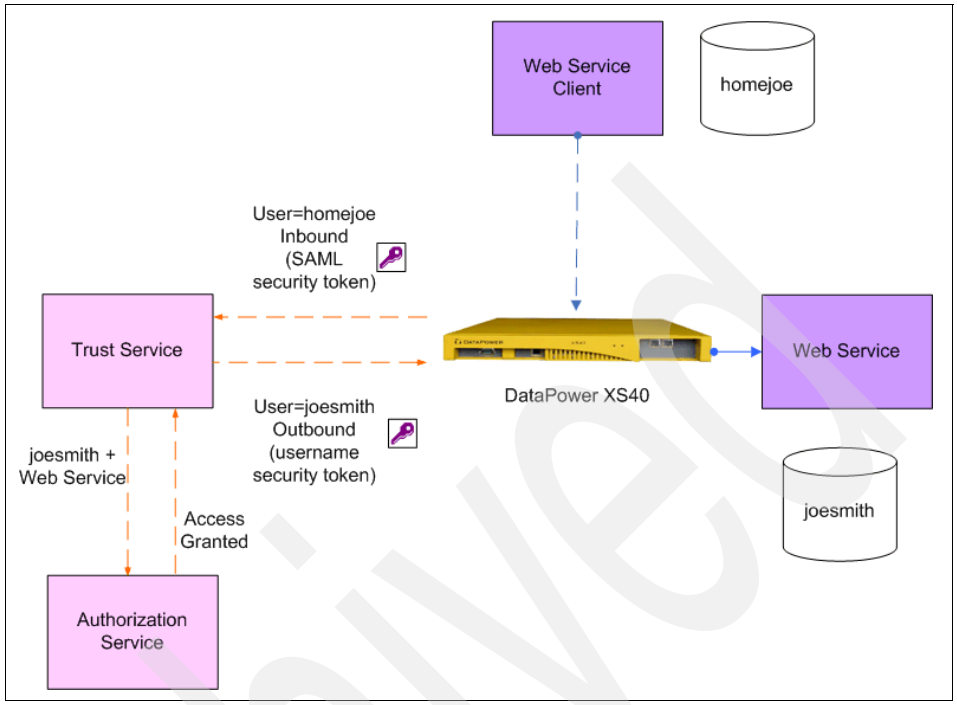


Figure 24-29 DataPower XS-40 as Web services gateway with authorization

For a thorough discussion on the Web services gateway architecture pattern, including how the gateway is used on the service requester side, read section 26.2.3, “XML gateway pattern” on page 828.

24.4.4 WS-Trust

The WS-Trust specification defines the interface used to manage the security tokens defined by the WS-Security specification. The Tivoli Federated Identity Manager Trust Service interface is defined by WS-Trust. It may be accessed by trust clients using either SOAP requests or direct JAVA API calls. The trust client can be the one in Web services security management, SPS or a custom client as long as it conforms to the Tivoli Federated Identity Manager WS-Trust profile. This interface allows any conferment Trust Client to request security tokens from the Tivoli Federated Identity Manager Trust Service, where the Trust Service can provide the appropriate token translation, identity translation, and request authorization as part of its token functionality. For more on the Trust Service see 24.2.4, “Trust services” on page 729.

24.4.5 Authorization services

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

24.4.6 Web services security management architecture approach

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

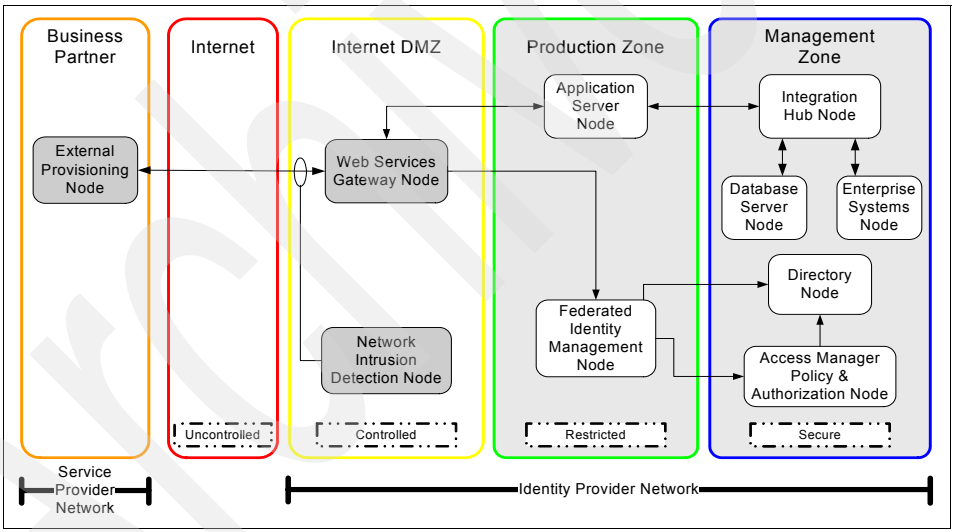


Figure 24-30 Specified level view of Web services security management

A more detailed look at Web services security management deployment is available in 26.2, “Federated Web services architecture patterns” on page 824.

24.5 Provisioning services

Provisioning services are used within a federated environment for both a prior and run-time provisioning solutions, as described in 23.6, “Federated identity provisioning” on page 716. Provisioning services interact with both local identity management systems (such as Tivoli Identity Manager) and local data stores (access via identity services). Provisioning services are leveraged to federate local identity management systems across federation business partners and to provide federated management of identity data, including transactional and profile attributes, see 23.2.5, “Identity attributes” on page 691.

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

WS-Provisioning describes the APIs and schemas necessary to facilitate interoperability between provisioning systems and to allow software vendors to provide provisioning facilities in a consistent way. The specification addresses many of the problems faced by provisioning vendors in their use of existing protocols, commonly based on directory concepts, and confronts the challenges involved in provisioning Web services described using WSDL and XML Schema.

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

Figure 24-31 shows the communications and exchanges that take place at each layer of Federated Identity Manager when performing Web services Provisioning.

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

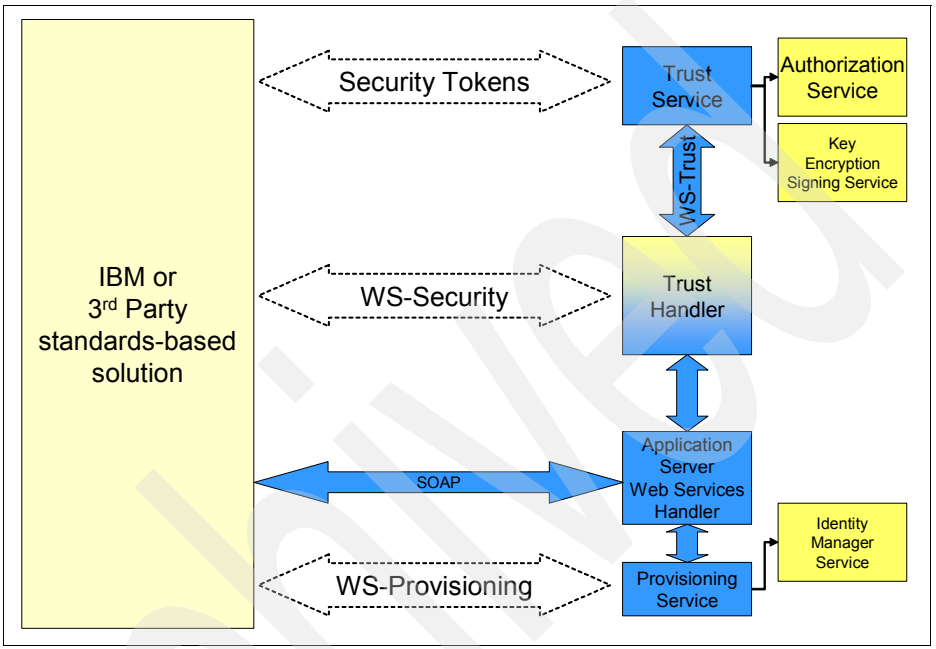


Figure 24-31 Web services Provisioning: Components and communication

At the Communication layer, SOAP messages are being handled by the Application Server, in this case WebSphere Application Server or WebSphere Web services Gateway. All real communication is via the Web services handlers in the Application Server.

At the “Protocol” layer, the WS-Security header in the SOAP request are handled by the Tivoli Federated Identity Manager Trust Handler. It must read the WS-Security headers sent by the third-party solution (incoming) or include headers for the third-party solution (outgoing).

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

SOAP Security is used to protect WS-Provisioning messages and the Provisioning service acts as a secured Web service, accessing the IBM Tivoli Director Integrator in the back-end.

24.5.1 Architecture overview

Figure 24-32 shows the components required in order to implement secure, cross-enterprise, provisioning using Tivoli Federated Identity Manager.

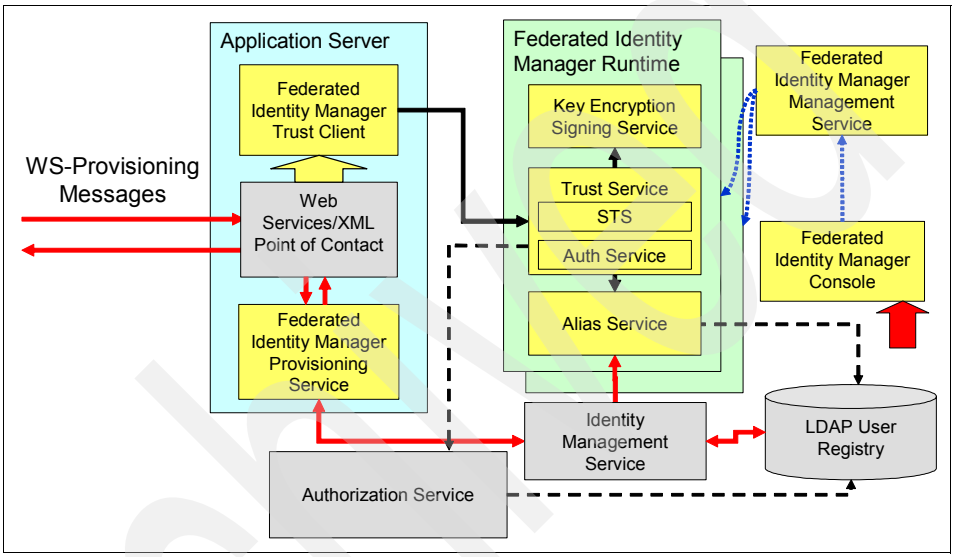


Figure 24-32 Components for federated user provisioning

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

The only components specifically related to provisioning are the provisioning service itself and Identity Management Service which is the enterprise Identity Management Service, in this case the IBM Tivoli Directory Integrator, but it could also be a bespoke identity provisioning capability. The Federated Identity Manager Alias Service and LDAP registry are also needed if provisioning for Liberty SSO with account linkage.

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

then passes it on to Directory Integrator. A Directory Integrator AssemblyLine extracts the identity information from the provisioning request and handles as appropriate. If the request is to provision a local account for Liberty SSO then the alias service is called to associate the newly created user with the received Liberty alias.

Although the diagram in Figure 24-32 on page 777 shows Directory Integrator interfacing directly to the LDAP user registry this is just an example. Directory Integrator could be configured to interface with any supported endpoint including IBM Tivoli Identity Manager.

Figure 24-33 provides an overview of the WS-Provisioning support provided in Federated Identity Manager. The Federated Identity Manager components are:

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

Both of these provide a full implementation of the three interfaces defined by the WS-Provisioning standard.

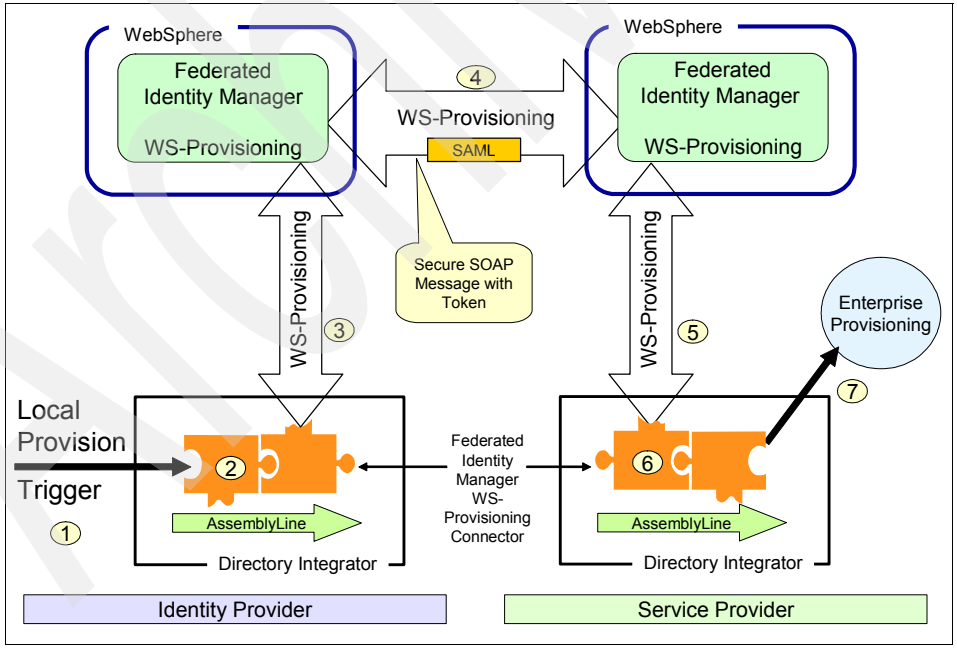


Figure 24-33 Federated provisioning - Overview

A provisioning event is sent from the identity provider to the service provider via this sequence:

1. Some type of *provisioning trigger* at the IP initiates an Tivoli Directory Integrator AssemblyLine. Tivoli Directory Integrator provides several mechanisms to start an AssemblyLine: the creation of a new entry in an LDAP directory is detected by a monitoring agent, a DSMLv2 request from Identity Manager or another enterprise provisioning service, and so on.
2. The Tivoli Directory Integrator AssemblyLine collects data to form a WS-Provisioning message. The AssemblyLine can use any of the standard Tivoli Directory Integrator facilities for this including the many standard Tivoli Directory Integrator connectors.
3. The Tivoli Federated Identity Manager WS-Provisioning connector sends a WS-Provisioning message to the Tivoli Federated Identity Manager WS-Provisioning Service.
4. The Tivoli Federated Identity Manager WS-Provisioning Service uses the Tivoli Federated Identity Manager trust server to create a SAML token for a configured identity and uses the WebSphere SOAP Security support to forward the WS-Provisioning message to the target service provider.
5. The Tivoli Federated Identity Manager WS-Provisioning Service on the SP receives the message and forwards it to a configured WS-Provisioning Connector on a local Tivoli Directory Integrator. This Tivoli Federated Identity Manager WS-Provisioning Service may be configured to use Tivoli Federated Identity Manager Web services security management for identity validation and request authorization with Tivoli Access Manager.
6. The Tivoli Federated Identity Manager WS-Provisioning Tivoli Directory Integrator connector receives the WS-Provisioning message and starts a configured Tivoli Directory Integrator AssemblyLine.
7. The Tivoli Directory Integrator AssemblyLine on the SP collects whatever local data is required and initiates local provisioning, using an enterprise provisioning system such IBM Tivoli Identity Manager if necessary.

Note that the WS-Provisioning messages sent between Tivoli Directory Integrator and Tivoli Federated Identity Manager do not include SOAP Security headers because they are assumed to be in a trusted environment. The WS-Provisioning messages from Tivoli Federated Identity Manager-to-Tivoli Federated Identity Manager do use the SOAP security support of WebSphere.

24.5.2 Provisioning architecture approach

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

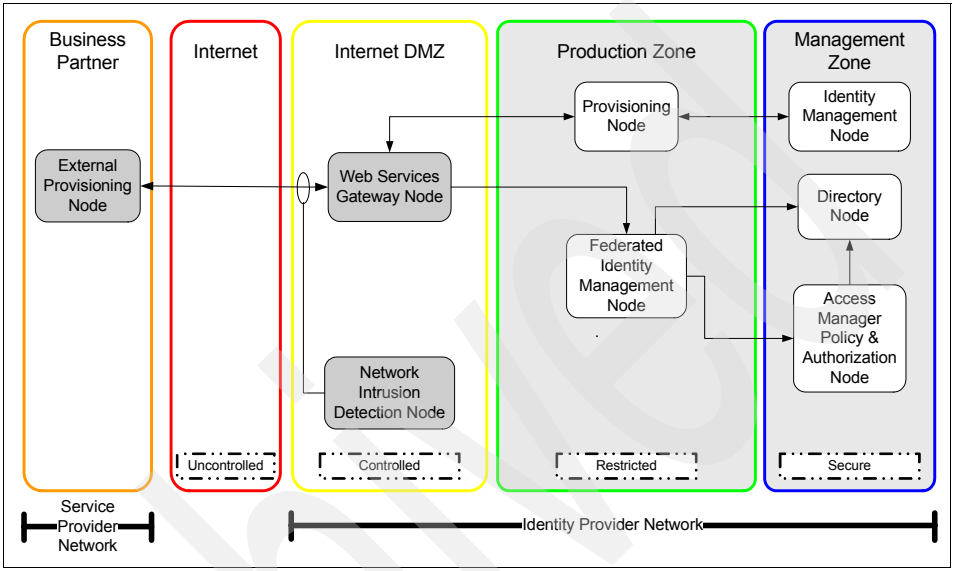


Figure 24-34 Generic IBM Tivoli Federated Identity Manager specified level view of provisioning

24.6 Conclusion

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

The focus of the chapter was to give a description of how the Tivoli Federated Identity Manager solution is implemented to meet the overall Federated Identity Manager challenge. We discussed how the Tivoli Federated Identity Manager product set is built around the trust infrastructure implemented by the trust service and provides point of contact servers for varying customer requirements. SSO services provide the implementation of federation protocols, and also the interface between the point of contact (PoC) and the trust service.

25

Cross enterprise federated single sign-on scenario

Federated single sign-on represents the process of providing single sign-on credentials to partner enterprises in order for users to have a seamless transition from one enterprise to the next or from one domain to the other.

The deployment characteristics of the enterprises involved mean that typically there are numerous methods of single sign-on that could be employed by security architects.

In this chapter we attempt to identify some of these integration opportunities and the benefits and challenges of each. We focus not only on integration between Tivoli Security products, but we also highlight a typical scenario of an enterprise implementing a standards-based federated security solution.

25.1 Business context

BankWithUs is a large bank. They have customers around the US, and as such, have many partner services to offer their customers. These partners manage such things as stock trading, credit card point systems, retirement investment management, to name a few. BankWithUs has 3.2 million registered online customers for their banking services.

Recently, BankWithUs conducted online surveys for their customers to discover how they can improve their services. One of the most common complaints from their customers was that they have to remember so many passwords when using BankWithUs' partner services. The argument generally was "BankWithUs wanted me to join the partner and even though they introduced me to them, I had to register independently and then remember *another* username and password."

To address this problem, BankWithUs engaged some of their partners in order to discover how best they improve their online services. The initial partners targeted include the following:

- StocksMustGain Corporation: StocksMustGain Corporation provides stock trading services to a set of BankWithUs customers. They also have their own independent customer base.
- PointsTech Corporation: PointsTech provides a purchase points system (similar to frequent flier points) to a small set of privileged BankWithUs customers with a particular type of business credit card.
- RetireNowPlease Corporation: RetireNowPlease manages retirement services for customers of BankWithUs. They also have their own independent customer base.

Currently, customers of BankWithUs maintain login identities and passwords for each of the partner sites that they want to access.

BankWithUs' customer base also includes customers that were introduced to them by either StocksMustGain or RetireNowPlease. Hence, BankWithUs saw an opportunity to minimize the cost of managing identities for the numerous partners, while increasing end-customer services and satisfaction.

BankWithUs corporation engaged with StocksMustGain, PointsTech, and RetireNowPlease corporations to gauge their positions to participate in this transformation. At the business levels, there is support between the organizations to enter into such an engagement. However, there is concern from the partners' management teams about the integration effort and whether they have the resources to successfully complete and support such.

As part of the business agreement, the next step was a technical appraisal for each partnership with the aim to satisfy the following requirements:

- Perform single sign-on for shared users within the solutions.
- Identify any identity management related cost savings for all parties.
- Provide the actions in the first two bullets without disproportionate investment in IT infrastructure or skills.

This chapter aims to provide conceptual designs for satisfying these core requirements.

25.2 Technical specifications

Each of the partners have differing levels of maturity when it comes to their online strategies. This section outlines the technical implementation of each corporation's online services as it currently stands.

25.2.1 BankWithUs Corporation

BankWithUs Corporation considers itself a leader in its evolution of online identity management, having established services for its customers in the late 1990s and extending this to include support for federated relationships with some of its business partners. Its SOA strategy is in progress, providing an access point for customers with active client user agents to leverage BankWithUs' applications. For providing external automated user provisioning services, BankWithUs implemented a WS-Provisioning solution to support real time provisioning requests. All of these strategies are underpinned by a security solution from Tivoli Security.

BankWithUs constructs their login identifies with first name, followed by "-" followed by surname. For example, John Howard would be provisioned a login identifier as *john-howard*.

Figure 25-1 on page 784 shows the BankWithUs corporation high level outline of components.

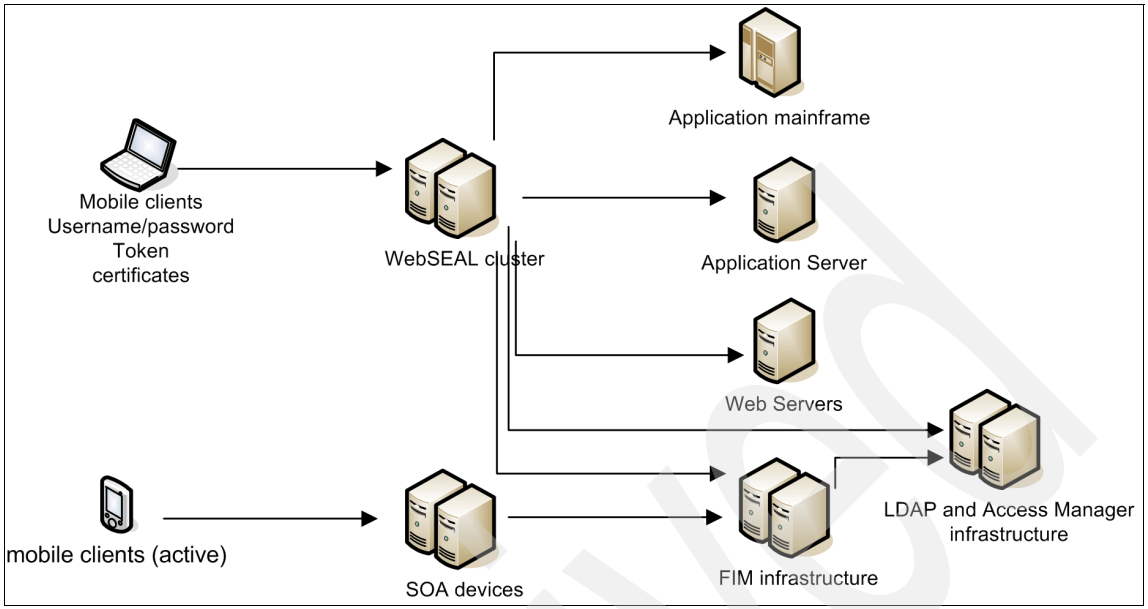


Figure 25-1 BankWithUs technical environment

As can be seen, the customer has a high maturity level from a security deployment perspective, with all the infrastructure in place to provide single sign-on to partners.

Functionally, from a security perspective, they are able to provide the following interfaces for customers to interact with:

- Tivoli Access Manager for e-business single sign-on options are outlined in 9.6, “Enterprise single sign-on mechanisms” on page 313. This includes the ability to provide support for a range of authentication mechanisms from the client.
- Federated Identity Manager SSO options outlined in section 26.1, “Federated SSO architecture patterns” on page 804. This includes the ability to act as an identity provider or a service provider in a federation relationship.
- Support for WS-provisioning requests through their SOA devices as outlined in 24.2.8, “Provisioning services” on page 735.

Let us now consider the partner’s implementations in order to determine the complexities for integration.

25.2.2 StocksMustGain Corporation

StocksMustGain provides online stock trading to Internet customers. They provide this service for customers of BankWithUs. At this time, initial estimates identified the customers targeted user population for integration to be around 100,000.

Figure 25-2 shows StocksMustGain corporation's deployment.

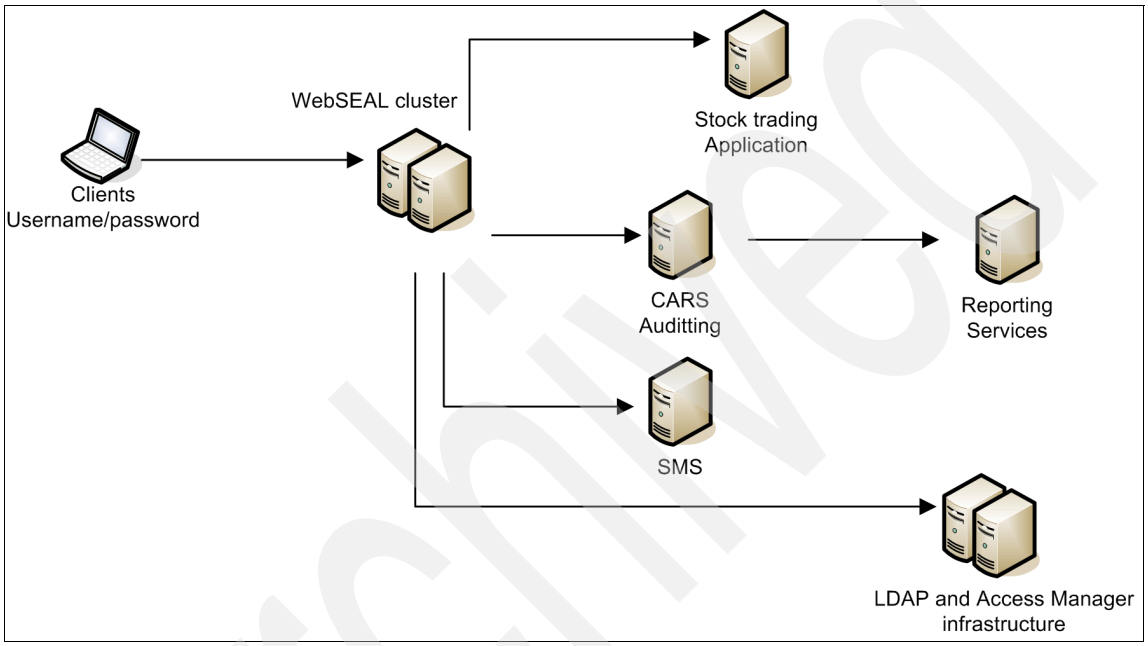


Figure 25-2 StocksMustGain technical environment

It shows a deployment of Access Manager, which protects the application infrastructure. This infrastructure includes stock trading services as well as other investment applications. StocksMustGain has a stringent audit requirement, hence, they extended Tivoli Access Manager for e-business to generate customized audit records through the Common Auditing and Reporting Service. For more information see Chapter 27, "Introducing IBM Tivoli Common Auditing and Reporting Service" on page 845. As shown, they also have a WebSphere cluster deployed that implements the Tivoli Access Manager Session Management Server to provide centralized session management.

StocksMustGain provisions login identifiers with first name, followed by an underscore (_) character, followed by surname. So, a user with the full name of John Howard is constructed as *john_howard*.

Note: StocksMustGain is in the process of trying to attain funding for building an SOA environment, including a Web services presence. Unfortunately, at StocksMustGain, there are tight budget controls in place, meaning acquiring funds for new IT projects is difficult, so this strategy is stalled somewhat.

StocksMustGain provides a number of applications to their customers, the most significant for this integration is the WebSphere Application Servers, which host the stock trading services.

25.2.3 PointsTech Corporation

PointsTech Corporation won the business to host the credit card points system for BankWithUs' privileged business credit card customers in late 2005. BankWithUs does not consider its point system a revenue stream, hence their focus was to out source and focus on its revenue generating core banking services. The concurrency and service level requirements for the application are low. Redundant infrastructure was not required in order to support the service level requirements of BankWithUs.

Customers accrue (build up) points by spending money using the bank's platinum credit card. Once they have enough points, they can be used to purchase rewards. The application provides customers a way to view their points balance online. At this time there is no automation that allows purchases to be made using these points online.

In addition to the customer view of the application, some management services are provided for PointsTech help desk personnel for them to process purchases made with points by the customers. All reward purchases are made via a free call number to PointsTech Corporation.

PointsTech built the required application, and has been running successfully since. PointsTech has dedicated hardware and software for BankWithUs' customers. The initial customer set is 30,000 customers in total, with a low concurrency requirement. The application is the only Web-based application that they host externally. Their decision to deploy a simple infrastructure that uses Microsoft IIS with an inhouse developed .NET application. Active Directory holds the user records.

Figure 25-3 on page 787 shows PointsTech Corporation's architecture.

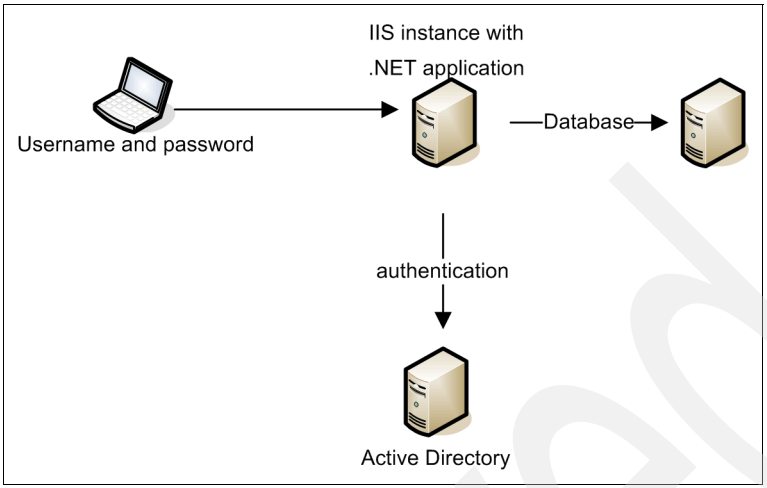


Figure 25-3 PointsTech's deployment environment

As can be seen, the security design is quite simple, with an IIS .NET infrastructure deployed to support this one application using Active Directory as the authentication identity store. The database is uploaded nightly with points and identity information from BankWithUs. As part of this process, Active Directory is loaded with new users' details as subscribed by BankWithUs. During this upload, a password is generated and e-mailed to customers for initial login to PointsTech.

PointsTech has no strategy for Web Services Support but is being driven by BankWithUs and other partners to create a plan for doing this. Currently, identity management is performed by their Help Desk staff.

25.2.4 RetireNowPlease Corporation

RetireNowPlease provides retirement investment services (superannuation, 401K, and so on) for customers of BankWithUs. Initial estimates of a shared customer set reach 350,000; however, this number may expand after RetireNowPlease and BankWithUs advertise the capability of performing single sign-on to its customers.

RetireNowPlease corporation embraced open standards. They invest in open source solutions and rely on their internal staff to support and build required functions for deployment. This model means that they are generally slower to adjust to change. It also means that from a support perspective, they must be self-sufficient. As with their infrastructure, their security solution is developed internally.

RetireNowPlease constructs user identities based on a user's account number.

Figure 25-4 shows RetireNowPlease's conceptual architecture.

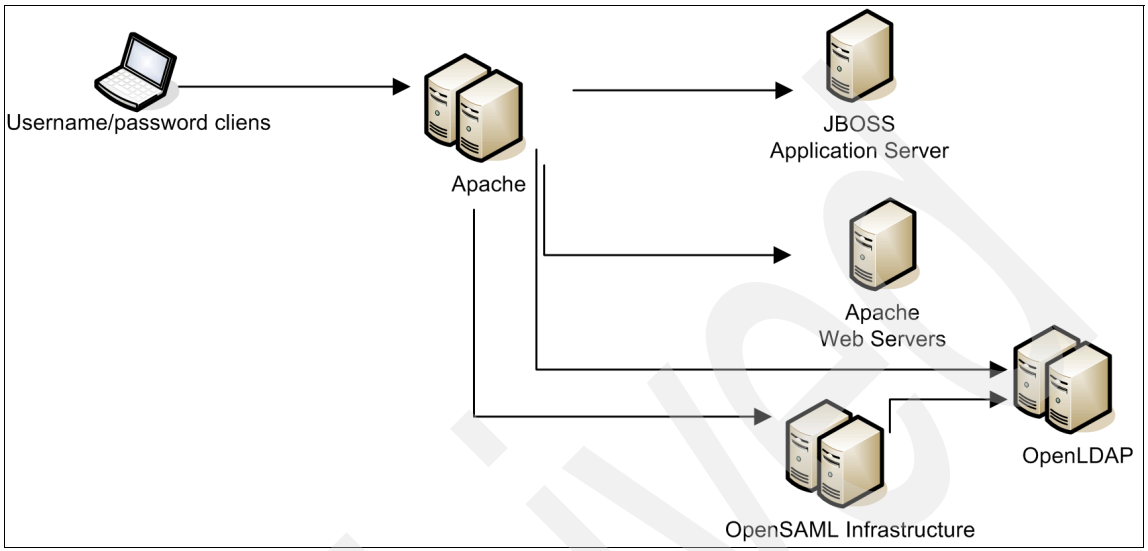


Figure 25-4 RetireNowPlease's technical environment

RetireNowPlease uses the OpenSAML source tool-kit as their federated single sign-on solution with openLDAP as the directory of choice, which you can find out more about at the following Web sites:

`http://www.opensaml.org`
`http://www.openldap.org`

The application is hosted by Apache with the mod-proxy plugin. The JBoss application server is used to host the single sign-on application, with Apache Web servers hosting the financial applications. Learn more about The JBoss application server at the following Web site:

`http://www.jboss.org`

Given the open source approach, BankWithUs expressed some concern with linking their customers with the RetireNowPlease Web site. BankWithUs corporation was reassured by the management of RetireNowPlease that every effort is made to minimize vulnerabilities within the software solution. As part of this commitment, RetireNowPlease's applications are subjected to the same penetration tests that BankWithUs performs regularly on their own applications.

The following sections show the technical solutions for integrating each of the partner sites with BankWithUs Corporation. Having understood the business

requirements and current infrastructure as previously outlined, the solutions vary based on both of these variants.

Note: The figures representing the organizations described above are simplified in the following sections in order to eliminate details that are not relevant within the configuration. For a full outline of the corporations' deployment, please refer back to this section.

25.3 BankWithUs engages PointsTech

BankWithUs' technical infrastructure and security specialists engage with PointsTech's technical specialists to determine how to address the business requirements. They understand that this change of requirements will cause additional up front costs, but they hope to make up that initial cost in on going reduction in identity management.

On the other hand, PointsTech has limited technical experience and are concerned about having to manage an enterprise security product deployment. Their desire with the solution is that it does not tie their implementation down to a single vendor solution.

PointsTech also does not own identities, nor do they have permission from BankWithUs to target any promotional or advertising content at their customers. Hence, they feel somewhat burdened by having to store and manage credential information for the site's users. They prefer to be able to just inherit some trust information from BankWithUs and use header content to identify the user.

25.3.1 Design decisions

The design considerations are as follows:

- Given there is no enterprise security solution deployed at PointsTech and no desire for such, a federated model is considered.
- Given BankWithUs is a Tivoli customer and has an awareness of the software capabilities of Tivoli Federated Identity Manager Business Gateway, this software is proposed to provide the federation management.
- SAML 1.1 is to be chosen as the specification. The additional features of SAML 2.0 were considered overhead for the simple requirements of single sign-on. For more information about the different protocols, read section 24.3.5, "Federated single sign-on approaches" on page 753.
- The customer is running an IIS infrastructure, with a .NET application, and will be a service provider in any federation relationship.

- Browser Post artifact will be used to remove the dependency on the WebSphere Cluster at the service provider.
- All users would be forced to perform single sign-on through the BankWithUs, removing the authentication requirements at PointsTech.
- No user lifecycle management processes (for example, change password) will be performed at PointsTech.

Figure 25-5 shows the conceptual design.

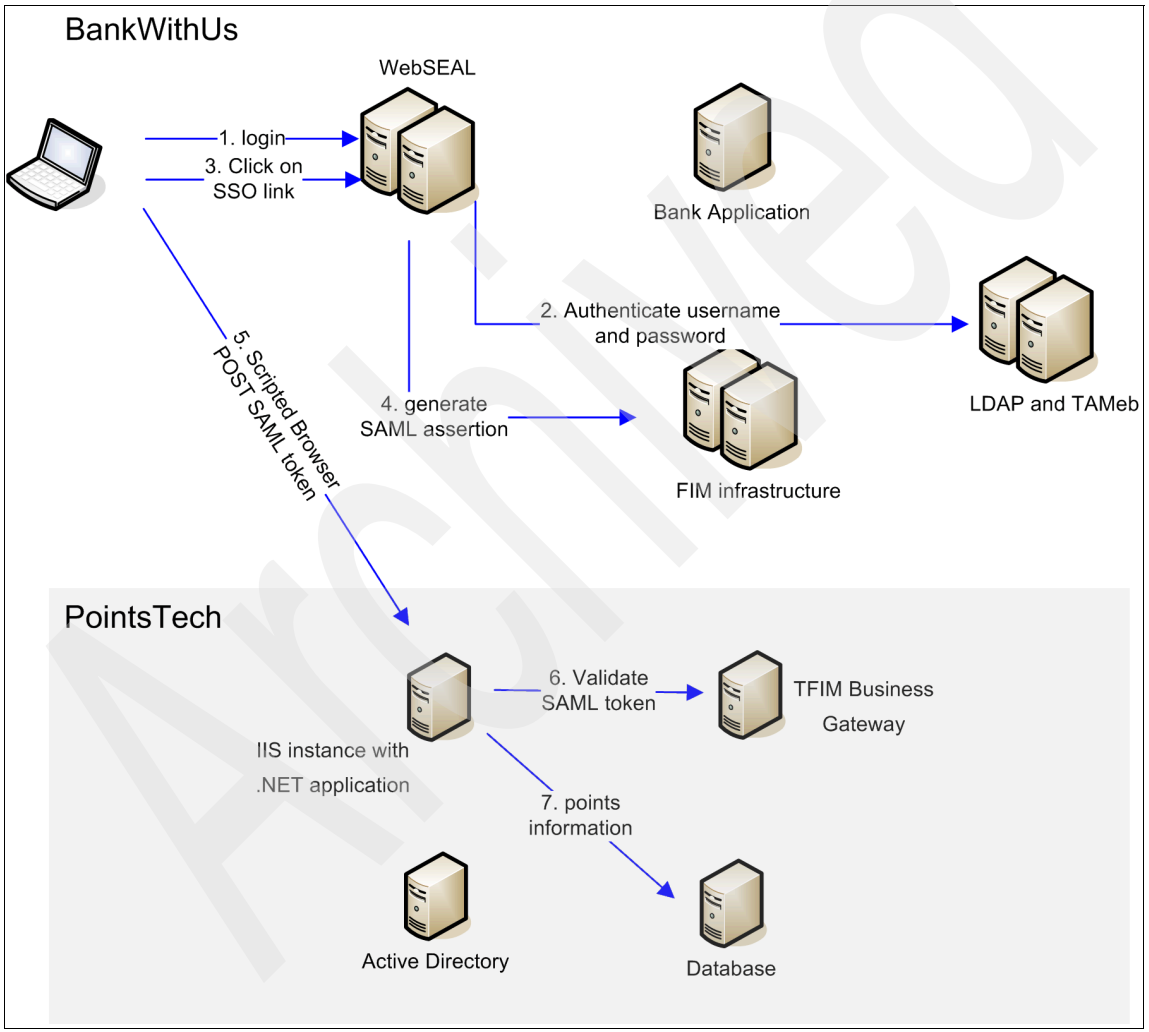


Figure 25-5 PointsTech new deployment environment

The following sections outline the scope of changes at each of BankWithUs and PointsTech.

25.3.2 Changes required

Let us shed some light into the required changes for both BankWithUs and PointsTech.

BankWithUs

There are no infrastructure changes for BankWithUs. Changes require a new federation partnership within Federated Identity Manager.

The following configuration changes are required in order to satisfy the business requirements:

- A new Federation needs to be configured. The identity provider will be BankWithUs. The service provider will be PointsTech.
- New keys for the federation need to be created and deployed to each partner.
- Identity mapping will be configured on the identity provider to map all users to a single user, adding the actual PointsTech identity as a SAML attribute.

The following application changes are required in order to satisfy the business requirements:

- The online application will require a new link to the single sign-on protocol browser post endpoint.
- Delinking of the federation will take place as part of deprovisioning of the actual credit card. Customers will not be given the option to have a login identity and password at both sites.

As far as process changes required, a general awareness for dealing with customers with linked accounts needs to be created. There is no real identity management or Help Desk processes that need updating. The expectations are that customers may generate Help Desk calls early in the deployment stage to query about the changes in user experiences. The process of transferring points information to PointsTech does not change.

PointsTech

The following infrastructure changes are required to support the business requirements:

- New hardware will be deployed with WebSphere Application Server (single server) to host the Federated Identity Manager Business Gateway.

- Federated Identity Manager Business Gateway software will be configured on the new hardware.

The new infrastructure is shown below, notice the infrastructure changes made in the PointsTech environment. The process of single sign-on is shown in the steps highlighted.

The IIS instance now has an additional Web server plugin that communicates with the Business Gateway when an incoming token is provided. Customers can now login once to the BankWithUs corporations Web site and single sign-on to the PointsTech application in order to view their points online.

The following configuration changes are required to support the business requirements:

- Federated Identity Manager Business Gateway software will be configured on the new hardware.
- The Web Server Plugin for IIS will be configured for the Business Gateway, hence it will be configured as a service provider.
- Federated Identity Manager Business Gateway needs to be configured as the service provider partner to BankWithUs.

The following application changes are required to support the business requirements:

- A header identifying the user is passed to the downstream .NET application. A generic user is used as the authenticated context with IIS and Active Directory.
- Identity Management processes for outside users is disabled at PointsTech. All users come into PointsTech via the BankWithUs channel.
- For help desk users, the .NET application is configured to challenge the user to provide authentication credentials via username and password.
- Help Desk staff focuses on points redemption rather than identity management. No identity management Help Desk routines will exist.

The following process changes are required to support the business requirements:

- Help Desk staff is educated about the new function scope within the application.
- Processes for identity loading of account information into Active Directory will be eliminated. All Internet customer accounts are deleted from Active Directory.

25.4 BankWithUs engages RetireNowPlease

BankWithUs and RetireNowPlease both compete in the finance industry and provide complimentary services for some individuals.

The organizations often run joint advertising campaigns on each other's Web sites in order to attract business from their partner's user base. Typically, customers with a relationship with BankWithUs request retirement account services with RetireNowPlease. Conversely, although less frequently, customers of RetireNowPlease wish to open accounts with BankWithUs.

Both corporations like to think that they have established strong relationships with their customers, and see this relationship as being a key driver for new cross-enterprise business opportunities.

For both organizations, managing their large user base is costly. Those customers with accounts at both organizations require each organization to keep different username and passwords for a single customer. From a customer's perspective, this does not make much sense, as the driver for having to acquire an account at one corporation often comes from the other corporation (through advertising for example). One comment by a customer in the recent survey is quoted as saying:

"BankWithUs wanted me to sign up for RetireNowPlease services, and now I have to manage user-name and identities at both corporations. Surely there is some way that you could collaborate to allow me to continue my strong and trusted relationship with BankWithUs. This situation seems ridiculous to me."

From a technical perspective, their approach to open standards adoption is similar on both sides, although their philosophy towards technical implementation is different. From a security perspective, they both implemented solutions that provide SAML 2.0 support for federated single sign-on. BankWithUs implemented Tivoli Federated Identity Manager, and RetireNowPlease uses open SAML as a toolkit implementation.

Let us now look at some of the integration decisions and the resulting benefits and challenges for each corporation.

25.4.1 Design decisions

The technical team started out by analyzing the two corporation's user population and found approximately 25% of all users possess login identities at both sites. This is more than they had expected. If federated, this presents a potentially large cost savings for each organization.

Having performed this analysis, and coupling that with the business requirements presented earlier, the following design decisions were made:

- Both BankWithUs and RetireNowPlease will be identity providers.
- Both BankWithUs and RetireNowPlease will be service providers.
- Both BankWithUs and RetireNowPlease will use SAML 2.0 as the protocol definition, allowing for full functional exploitation of available standards.
- All users will be given the option of choosing to federate to the service provider partner when they login to either site. After a choice is made, the user will not be asked again, but will be given the opportunity to defederate at any time.
- Identities will be referenced by aliases between the two sites, mapping rules will be developed.
- Both BankWithUs and RetireNowPlease will provide WS-Provisioning support from their partner.

Given that each site is similar in its standards support, no infrastructure of software deployment changes need to be forced by either party.

25.5 BankWithUs engages StocksMustGain

BankWithUs' technical deployment and security specialists engaged with StocksMustGain to analyze their environment characteristics, including the identity population. Given that they are both Access Manager deployments, there is functionality available within the Web security servers that they can both take advantage of in order to perform the enterprise single sign-on.

25.5.1 Design decisions

That being said, a number of design choices were made:

- Access Manager core features would be used to perform the single sign-on. This will satisfy the budget constraints of StocksMustGain.
- Best effort would be performed in order to provide seamless identity lifecycle management. A mix of automated processes on top of the Help Desk will provide the service desk requirements.
- This solution is tactical, rather than strategic for StocksMustGain. They are committed to providing a migration path to open standards beyond this project.

Following are the specific product features required in order to satisfy these business requirements:

- Cross Domain Single Sign-On, as described in 9.6.1, “Cross Domain Single Sign-On” on page 314, is used as the cross enterprise single sign-on mechanism.
- A custom Cross Domain Mapping Function module, as discussed in 9.6.3, “Cross Domain Mapping Framework” on page 321, is written in order to map the login identifier between the two different formats. This code should be relatively straight forward as the mapping is static for all users.

The high level view of the infrastructure is shown in Figure 25-6 on page 796, with the steps outlining the user interaction with the Web site and initiating a single sign-on operation. As can be seen, no FIM infrastructure is in use through this integration.

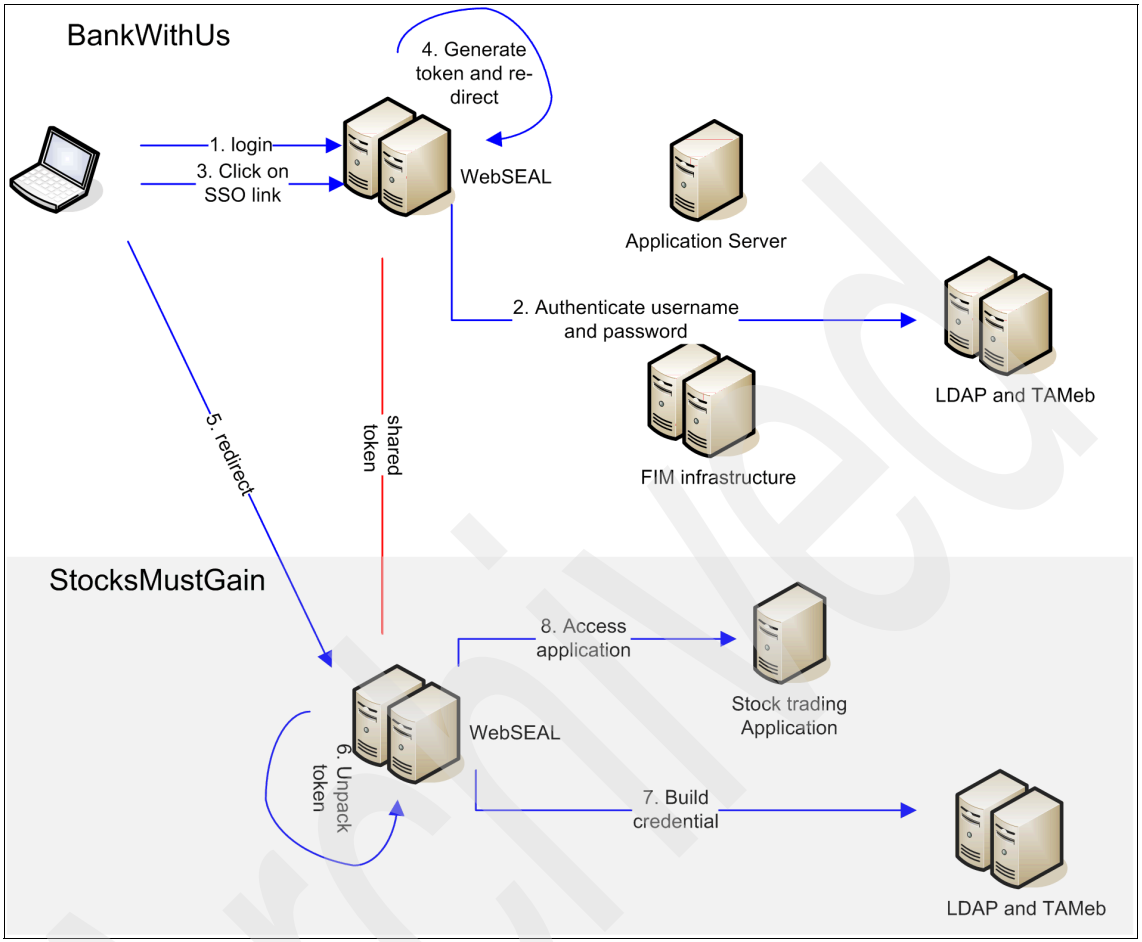


Figure 25-6 BankWithUs new technical environment

Let us look at the changes required by both BankWithUs and StocksMustGain in order to support this solutions.

25.5.2 Changes required

This section outlines in more detail the changes required in the infrastructure of both BankWithUs and StocksMustGain in order to satisfy the requirements.

BankWithUs

The following configuration changes are required in order to satisfy the business requirements:

- The WebSEAL servers at BankWithUs need to be configured to token generation for secure HTTP only. The consumption configuration, as well as the mapping, would be done at StocksMustGain.
- Each customer's stored identity information would be extended to capture linked identity status.
- A one-off, customer merge would be performed to mark certain users as potential targets of the single sign-on features.

The following application changes are required in order to satisfy the business requirements:

- On first login to BankWithUs, users will be asked to accept terms and conditions of the single sign-on operation. Those not accepting the terms and conditions will retain login identifiers at both sites. BankWithUs' customized inetorgperson object will be used to retain the migrated status of a user.
- Identified StocksMustGain stock users would be provided a "Login to StocksMustGain stock services" link next to the logout button on the Web site. Clicking this link will enable the single sign-on operation to occur.
- Only authorized users would be able to access the link (it would not be shown to all users, as well as providing runtime authorization if activated by a user).
- Help Desk staff will need additional functions to handle the different workflow processes around the user lifecycle.

The following process changes are required to support the business requirements:

- Cryptographic keys shared between BankWithUs and StocksMustGain will be rotated every 30 days.
- Help desk operations will need to be extended to include the new processes around managing the linked identities.

StocksMustGain

The following configuration changes are required to support the business requirements:

- The Cross Domain Mapping Function will be configured in the WebSEAL servers configuration file.
- LDAP inetorgperson object will be extended to capture linked status.

The following application changes are required to support the business requirements:

- Logout function will cause a local logout, and then redirect the user back to BankWithUs' Web site.
- The ability for a customer to opt-out of the federation will be given.
- Password management operations will be removed for those customers who have linked their accounts.
- Customer's with linked accounts who attempt to login locally will be redirected to BankWithUs' Web site on failure. BankWithUs will present an appropriate login page.

The following process changes are required to support the business requirements:

- All linked accounts will have their local password un-set so they cannot login locally.
- Help desk operations will need to be extended to include the new processes around managing the linked identities.
- For handling the update of the cryptographic token every 30 days, new processes need to be managed.

Ultimately, this solution is a tactical solution that will help to satisfy the short term requirements of single sign-on, but strategically, it is within StocksMustGain's plans to engage with Tivoli for acquisition of the enterprise Federated Identity Management product. This will help with the automation of the identity lifecycle management processes as well as extend their ability to act as identity providers and service providers for other scenarios.

Note: As part of Access Manager for e-business, e-community single sign-on is a natural succession of the CDSSO functionality. A use case is not presented in this chapter since, in our experience, most customers solve the same problem today with Federated Identity Management solutions. The open standards implementations provide a much richer feature/function set when compared to e-community single sign-on; hence, many customers choose this approach as opposed to this proprietary solution offered by Access Manager for e-business.

25.6 Benefits and challenges

Clearly, overall, the objects identified earlier, of providing single sign-on between BankWithUs and it's partners while keeping simplicity at a minimum has been

achieved. These objectives created some challenges along with some benefits within each organization, which this section attempts to identify.

25.6.1 BankWithUs

BankWithUs has a lot to gain from these integrations. Although the previous sections establish a one-to-one relationship with each of its partners, when put together, the result is a single authentication provider for many customers. Customers with relationships with each of the service providers can choose to federate *all* of the service providers under the BankWithUs brand.

Hence, BankWithUs can promote itself as an identity provider to all its user population, providing online services to them ranging from stock servers, to retirement investment services, to online points management servers—all with the possibility of users federating their identity under a single umbrella. This has the potential to strengthen its brand recognition to its customers and beyond, as well as to create growing satisfaction to its customers.

Along with the potential benefits, there is the potential for cost savings in the identity lifecycle management of its users. Eliminating the need for manual identity management processes with its partners has the potential to reduce costs.

The challenges for BankWithUs is to make sure that this maturity is enabled within its partners. As a leader in federated identity management, it has a responsibility to assist these smaller business partners in building a standards-based solution. As new standards are ratified and deployed within BankWithUs, there will be ongoing cost reduction for moving into automated processes around identity management, and ensuring that their partners are in a position to move with them will help realize these savings.

25.6.2 StocksMustGain

StocksMustGain have fulfilled their requirements to provide a single sign-on solution for its customers, while not requiring additional hardware or software cost.

The challenge for StocksMustGain is to break free from the cost reductions, and try to bring their infrastructure up to a maturity level that will integrate more freely with other partners. At the same time, they have the potential to become an identity provider to other partners and up the potential to become an identity provider, assisting its service providers with seamless single sign-on integration and reduced identity management. Giving their established customers the opportunity to solidify their relationship by nominating StocksMustGain as their identity provider will strengthen their brand.

25.6.3 PointsTech

The up front cost for PointsTech is minimal and does not require deployment of an enterprise security solution. PointsTech is now in a position to provide an open standards server provider interface to other customers.

Given the increased complexity of the infrastructure, additional management costs were introduced. This might consume some of the cost savings from the reduced management.

25.6.4 RetireNowPlease

After being presented with these proposed changes, RetireNowPlease asked for an extended period of time to implement the capabilities. As is expected of a toolkit approach, the time to market is longer than that of an off-the-shelf product implementation of the standards. This is an on-going challenge for BankWithUs, since the Tivoli Federated Identity Manager product requires simply a definition of the federation attributes in order to create a relationship.

As with BankWithUs, the benefits of this integration are vast. The user experience improvements, allowing a customer to single sign-on from their chosen identity provider, are significant. Allowing the user the opportunity to maintain their relationship with their chosen identity provider and to use the RetireNowPlease site as a launchpad for other services is positive. From the corporations point-of-view, not having to manage shared password and identity information for all users is a step forward, translating into reduced Help Desk calls and therefore reduced maintenance cost.

Along with this, the ability to automate their identity management processes through WS-Provisioning removes many of the manual processes that are followed by the help desk. Again, a reduction in costs is expected.

25.6.5 Customer

The customer has a lot to gain from these integrations. They have the opportunity to reduce their password management by taking up services offered by partners of BankWithUs. Having a single, trusted, identity provider gives the user peace of mind when performing online transactions, relying somewhat on the established business relationships between the providers to provide that comfort.

25.7 Conclusion

As can be seen, Tivoli Access Manager and Tivoli Federated Identity Manager product families offer a spectrum of single sign-on solutions. Couple this with the Tivoli Access Manager for Enterprise Single Sign-On product, as described further in Chapter 15, “Access Manager for Enterprise Single Sign-On” on page 449, and it simply becomes a design issue as to which products to utilize where. Hopefully, this chapter provides some guidelines as to which approach should be taken in a purely Web based federated single sign-on model.

26

Tivoli Federated Identity Manager patterns

Our earlier discussion of Tivoli Federated Identity Manager was helpful in describing the basic technologies, standards, and components of a federated identity management architecture. At this point, we apply those guidelines to a simple scenario involving a corporation that wants to establish a federation with a service provider.

This chapter describes the following:

- Architecture options for deploying Tivoli Federated Identity Manager and Tivoli Federated Identity Manager Business Gateway
- Approaches for integrating Tivoli Federated Identity Manager with other middleware and customer applications
- Several important issues relating to deploying Tivoli Federated Identity Manager in a production environment

26.1 Federated SSO architecture patterns

Tivoli Federated Identity Manager is a flexible product set that provides a federated identity management solution for both browser-based single sign-on and Web services environments. As there are many different examples of environments that require a federation solution, there are many different ways that Tivoli Federated Identity Manager can be deployed. We can represent the deployment of Tivoli Federated Identity Manager with several typical deployment/architecture patterns. In this section, we describe the most patterns from which customer specific deployments can be generated. The pattern for Tivoli Federated Identity Manager Business Gateway is shown in section 26.1.7, “SMB Pattern” on page 819.

26.1.1 Architecture approach

Tivoli Federated Identity Manager’s federated single sign-on (F-SSO) solution enables the single sign-on of a user in a cross-enterprise, or cross-domain, scenario. Tivoli Federated Identity Manager’s F-SSO functionality does *not* replace an enterprise’s existing authentication and session management services or any of the sign-on functionality they provide to the enterprise’s applications. Tivoli Federated Identity Manager’s F-SSO solution handles SSO to an edge-based *point of contact* component. This is based on the underlying principal that because Tivoli Federated Identity Manager does not replace existing session management functionality, it should not directly provide single sign-on to individual applications within an enterprise (enterprise single sign-on).

An architectural model based on a (scalable, available, performance) point of contact provides many security benefits, including the ability to control all access to an environment and closing off *back doors* that all unauthorized users use to access an enterprise’s environment. Typically an edge component, such as Access Manager for e-business, acts as a point of contact and is used to provide single sign-on from Tivoli Access Manager for e-business (where the user’s authentication credentials are collected and evaluated) to individual back end applications. This functionality remains unchanged by the addition of a Tivoli Federated Identity Manager solution. With the recent release of Tivoli Federated Identity Manager Business Gateway, the support for a broader range of point of contact servers was introduced, giving the customer greater freedom when making architecture decisions.

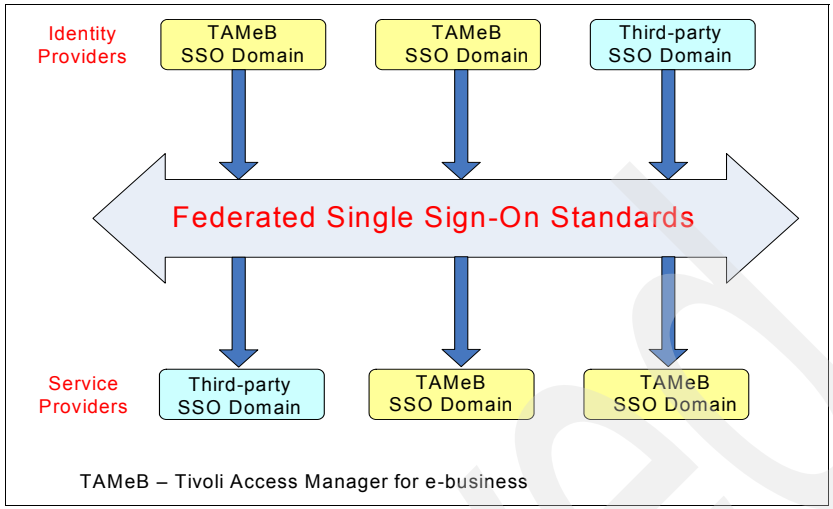


Figure 26-1 Linking SSO domains with Federated SSO protocols

This architectural approach to Federated SSO provides the following advantages:

- Little or no changes are required to enterprise applications
- Lightweight SSO within a domain
- Support for identity provider applications
- Choice of being able to leverage existing Tivoli Access Manager for e-business infrastructure

Little or no change to applications

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

The Tivoli Federated Identity Manager approach leverages Access Manager for e-business's ability to provide SSO to an application with little or no changes to the application. For those applications that use underlying middleware functionality to manage authentication, the middleware container can usually be configured to accept the user identity from Access Manager for e-business

without any changes to the applications using the authentication data. For example, IBM WebSphere Application Server provides the Trust Association Interceptor Plus (TAI++) to accept a user ID from an HTTP header variable, and create a login context for that user. Most other middleware products have similar functionality. For those applications that implement their own custom authentication logic, a small change to the login module to accept the user identity from a HTTP header variable rather than prompting the user for a user ID and password, is typically fairly straight forward to code and test.

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

The Tivoli Federated Identity Manager Business Gateway solution allows for a broader choice of deployment strategy by removing the dependency on Access Manager for e-business and providing support for IIS and WebSphere Application Server as point of contact solutions. Adding such capabilities has introduced the SMB pattern discussed in section 26.1.7, “SMB Pattern” on page 819.

Lightweight SSO within a domain

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

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

Support for identity provider applications

Even for pure identity provider deployments (no local services/protected resources are made available to the user), there are often self-care and portal applications associated with the identity provider’s identity management

functionality. The use of Access Manager for e-business to provide the *point of contact* for the identity provider leverages the (lightweight) SSO facilities of Access Manager for e-business to access the identity provider applications without incurring the overhead of running and accessing a separate service provider site for those applications.

Leverage existing Access Manager infrastructure

For those customers who already deployed an Access Manager for e-business SSO infrastructure, upgrading it to provide Federated SSO functionality is a relatively straightforward exercise. Moreover, in most cases the applications will not require any modification, thereby significantly reducing the time and costs needed to deploy the Federated SSO functionality. Tivoli Federated Identity Manager (as opposed to Tivoli Federated Identity Manager Business Gateway) precedes Tivoli Access Manager as the point of contact enforcement point.

26.1.2 Base pattern

The *Base architecture pattern* for deploying Tivoli Federated Identity Manager for Federated SSO uses the reverse proxy component of Access Manager for e-business (WebSEAL) to provide the *point of contact* for Tivoli Federated Identity Manager, namely authentication (at the identity provider side) and session management (for both an identity provider and service provider deployment). In this Base pattern, all users who use the Federated SSO functionality are individually defined in the Access Manager for e-business user registry[2].

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

If a user requests single-sign-on (or has it requested on their behalf by a service provider partner), Access Manager for e-business will pass control to the Tivoli Federated Identity Manager server. Note that Tivoli Federated Identity Manager itself, and the single sign-on functionality, can be access controlled by Access Manager for e-business. This has the effect of allowing a customer (in a more

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

advanced deployment) to provide single sign-on functionality to a subset of its users. Included with this request to Tivoli Federated Identity Manager will be the user's local (Access Manager for e-business based) identity. The Tivoli Federated Identity Manager server will use this identity for the building of the assertion provided as part of a single sign-on response.

Figure 26-2 shows the base pattern for an identity provider.

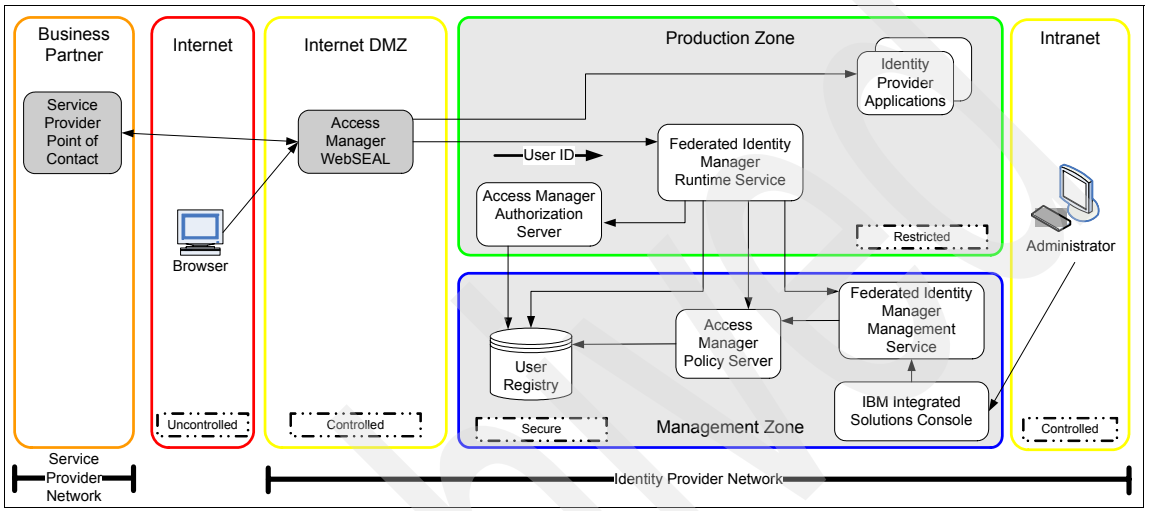


Figure 26-2 Base pattern for identity provider

For a service provider configuration, Access Manager for e-business (WebSEAL) is configured to allow unauthenticated access to the Tivoli Federated Identity Manager application, namely the login endpoint associated with the federation. After Tivoli Federated Identity Manager successfully validates and processes the incoming SSO message, it creates an Access Manager for e-business credential and passes it back to the WebSEAL server via the Access Manager for e-business External Authentication Interface (EAI). This allows WebSEAL to establish and manage an authenticated session for the user. See 9.4.6, "External Authentication Interface" on page 297, for a description of the External Authentication Interface of Access Manager for e-business.

Figure 26-3 on page 809 illustrates the base pattern for a service provider.

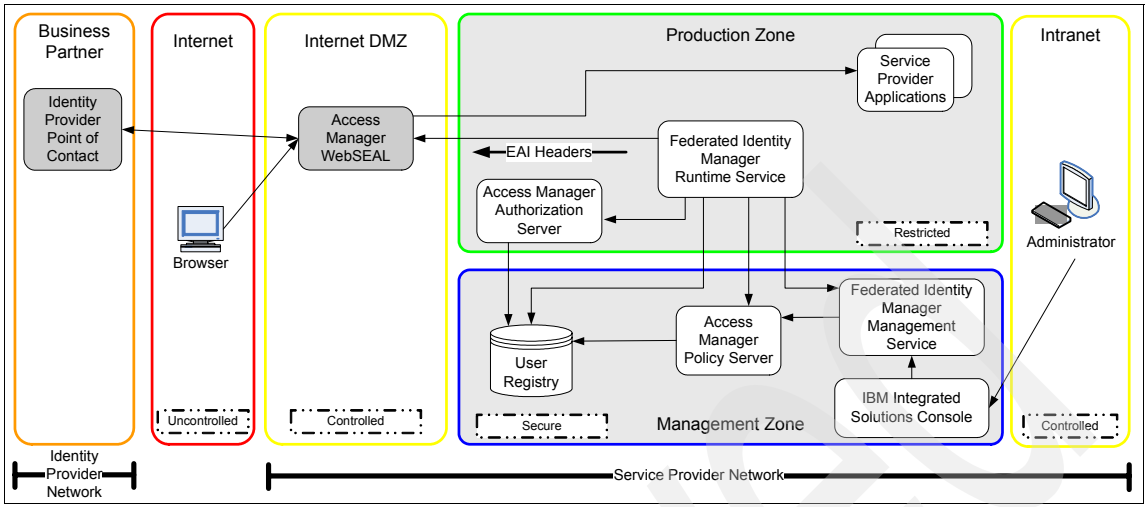


Figure 26-3 Base pattern for service provider

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

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

- Rely on message level authentication
- Rely on channel level authentication

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

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

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

Figure 26-4 shows the base pattern for an identity provider using Access Manager WebSEAL server to handle SOAP requests.

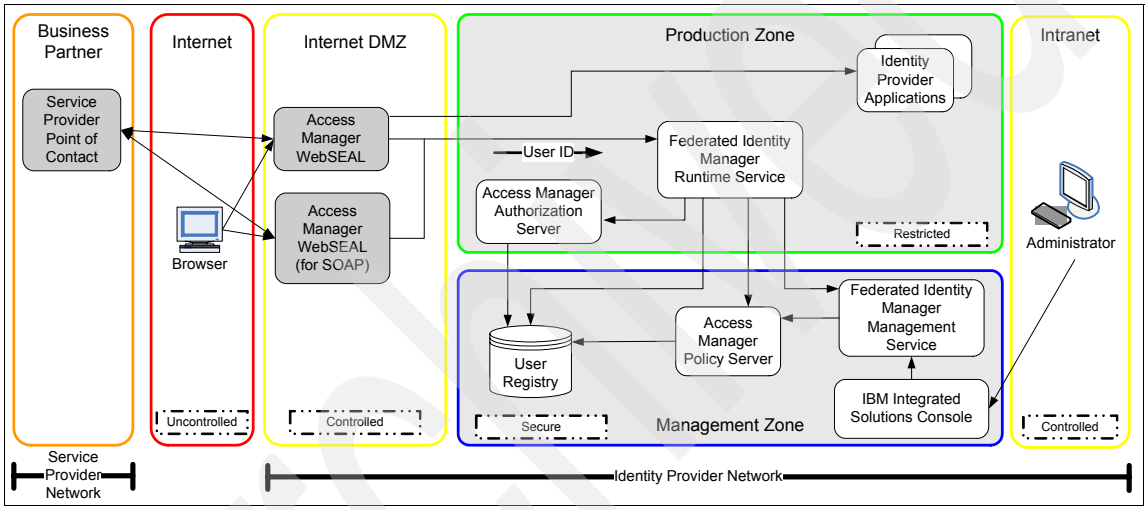


Figure 26-4 Base pattern for identity provider with SOAP back channel

26.1.3 Plug-in pattern

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

In this design pattern, the Access Manager for e-business Web server plug-in is configured into the Web server acting as a proxy for the WebSphere Application Server hosting the Tivoli Federated Identity Manager services. Access Manager for e-business-based SSO will be provided to any application running in the

same application server as Tivoli Federated Identity Manager. With Access Manager for e-business SSO, applications running on application servers in the same DNS domain can be implemented, but it requires the use of a domain cookie. Also an Access Manager for e-business plug-in must be installed in the Web server associated with the other application server(s).

Figure 26-5 illustrates the base pattern for an identity provider using the Access Manager Web server plug-in.

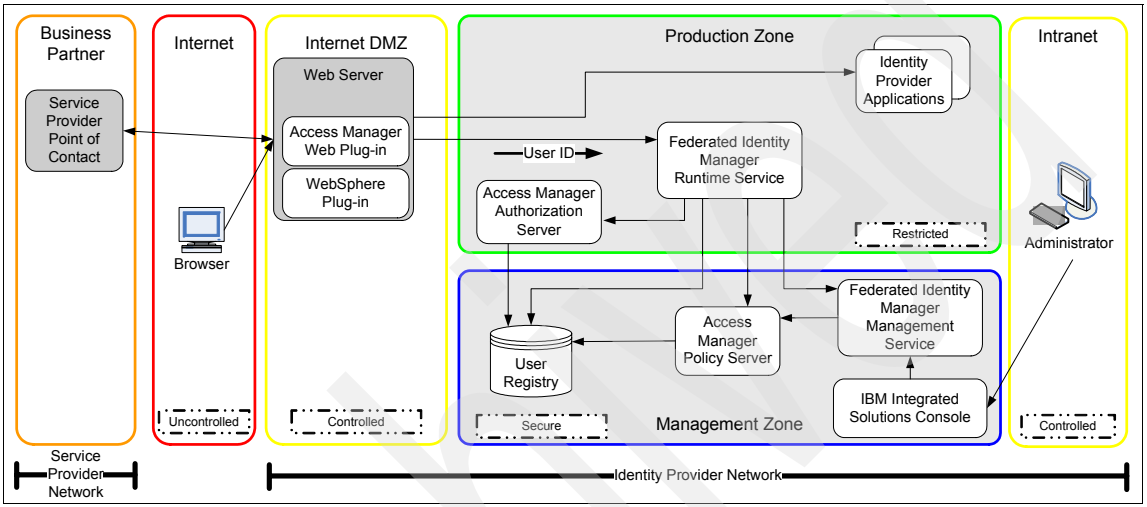


Figure 26-5 Base pattern for identity provider with Access Manager Web Plug-in

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

26.1.4 Lightweight Access Manager for e-business pattern

In certain cases, Tivoli Federated Identity Manager can be deployed using a light weight pattern for Federated SSO. In this pattern, Access Manager for e-business is leveraged for its session management capabilities *only*. Individual users are not stored in the Access Manager for e-business user registry and Access Manager for e-business related user management is largely done away with. Instead the Access Manager for e-business user registry contains either a single guest user ID or several role-based identities, with the identity mapping features of Tivoli Federated Identity Manager used to map to and from real user identities, as required.

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

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

On the identity provider side, the key to the Lightweight Access Manager for e-business pattern is the use of the External Authentication Interface (EAI) feature of Access Manager for e-business.

Example

Figure 26-6 on page 813 illustrates a sample light weight deployment of Tivoli Federated Identity Manager and Access Manager for e-business. In this light weight deployment, a user is authenticated against a Enterprise directory but does not exist as an Access Manager for e-business user within the Access Manager for e-business registry. This is significant because a user is (normally) required to exist within the Access Manager for e-business registry to allow Access Manager for e-business to build a local credential for the user. Recall that this credential is in turn used as part of the overall session management functionality provided by Access Manager for e-business, and so this credential is an integral part of Access Manager for e-business functionality.

In this light weight deployment, authentication is implemented through a custom login application. The Access Manager for e-business WebSEAL login page is redirected to this custom login application (Access Manager for e-business access control policy is defined such that this custom login application, and any images used are accessible by unauthenticated users). The custom login application displays a login page to the user and validates the user credentials entered by the user, using whatever method is appropriate for the particular deployment. In our example, the custom login application validates the user ID and password entered by the user against a custom user registry. The custom

login application is also responsible for handling any errors in login credentials entered by the user.

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

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

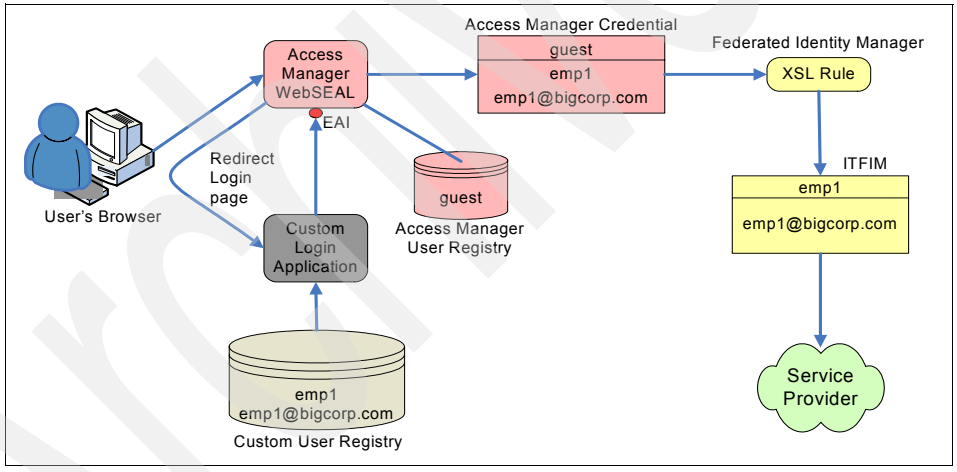


Figure 26-6 Example attribute flow for Lightweight pattern for identity provider

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

this guest user ID. It does not contain entries for each user identity that may be contained in an incoming SAML assertion.

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

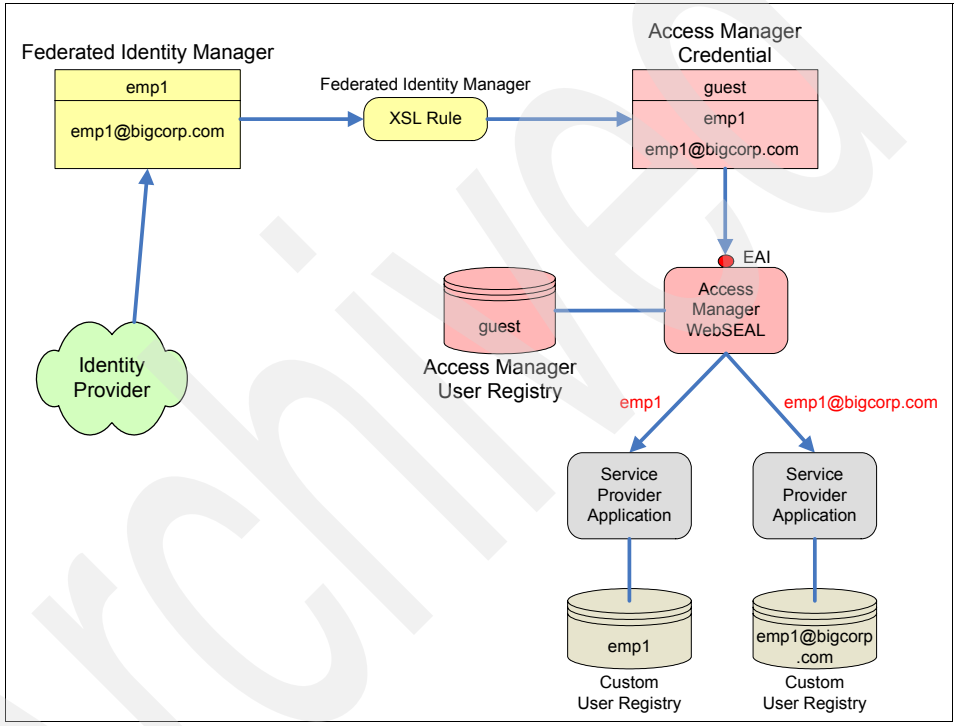


Figure 26-7 Example attribute flow for Lightweight pattern for service provider

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

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

26.1.5 Highly available architecture patterns

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

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

This scenario is depicted in Figure 26-8 on page 816.

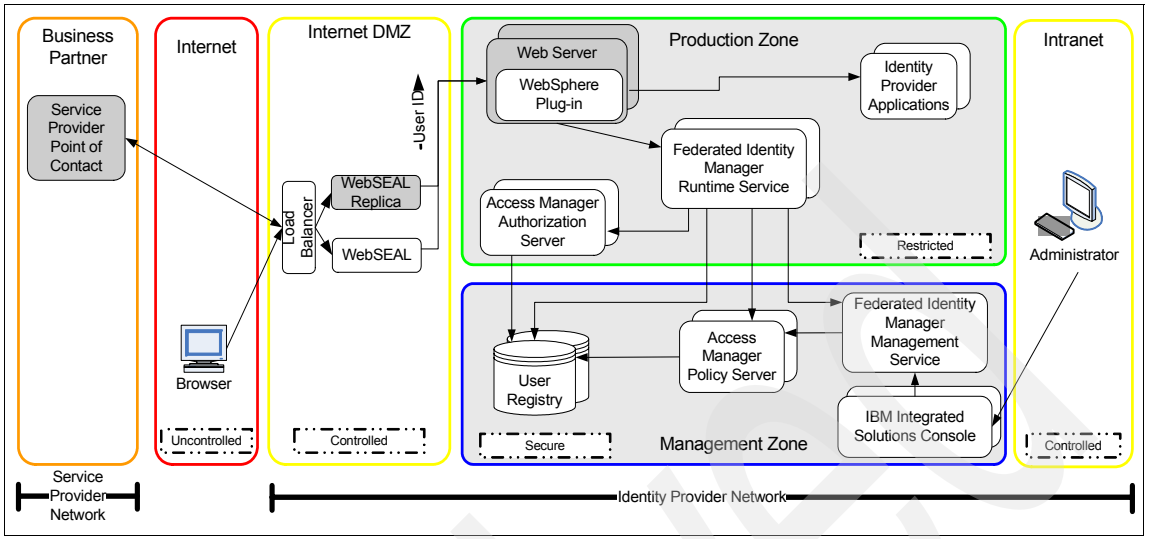


Figure 26-8 Clustered Base pattern

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

All Tivoli Federated Identity Manager Runtime nodes in a cluster use a shared session state, which is implemented using the DynaCache feature of WebSphere Application Server v6. This shared session state includes the assertion table for Browser Artifact profiles and contains sufficient information such that any of the nodes in the cluster can perform any operation. There is no need to ensure that subsequent operations for a particular Federated SSO session are directed to the same instance of Tivoli Federated Identity Manager Runtime. For example, one Tivoli Federated Identity Manager Runtime node may perform a Federated SSO operation, but any of the nodes in the cluster have access to the session

state information required to successfully perform a subsequent Single Logout operation for that session.

For a Federated Identity Manager Business Gateway solution, care must be taken when designing for high availability. Since the Runtime Services are supported on single server editions of WebSphere Application Server, cluster solutions are not permitted. This, however, does not restrict a customer from using multiple instances of such, but support for protocols that require cluster functionality, such as browser artifact profile of SAML, will not function as desired.

26.1.6 Multiple data center patterns

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

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

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

SAML 1.0/1.1/2.0

The configuration for SAML 1.0/1.1 depends on whether or not you are using the Browser Artifact profile for Federated SSO.

Browser Artifact Profile

If you are deploying the identity provider side of the Browser Artifact profile of SAML 1.0, we cannot solely rely on the *stickiness* of the WAN-based load balancing solution, as the Browser Artifact profile includes SOAP-based communication directly from the service provider to the identity provider. We therefore need to define a separate identity provider for each data center. The configuration at each data center will use different provider IDs and endpoint URLs even though they are logically performing the same role in the same federation. These provider ID and URL endpoint values use a logical host name that is unique to the data center. Service providers in the federation, regardless of whether they are implemented using Tivoli Federated Identity Manager, define a distinct identity provider for each data center.

Requests initiated from the browser to the identity provider (for example, via an SSO link from a browser page) can use the logical host name that the WAN-based load balancing solution was configured to balance across the data centers. The *stickiness* of the solution will ensure that subsequent requests after a Federated SSO operation will return to the same data center, and therefore be executed within the session state shared between the nodes at that data center.

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

Browser POST Profile

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

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

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

WS-Federation

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

Liberty ID-FF 1.1/1.2

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

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

26.1.7 SMB Pattern

Before the federation standards were accepted and later ratified, Tivoli Access Manager for e-business provided features that allowed Tivoli Access Manager customers to SSO between each other. This functionality, however, did not allow those other partners who were not Access Manager customers to federate into an Access Manager environment easily (notwithstanding the fact that other requirements such as SLO were not defined). Defining open standards helped overcome this. Now, any customer with a mature, standards compliant, vendor enterprise security solution (such as Tivoli) deployed can leverage that vendor's F-SSO implementation to federate to other enterprise partners.

That being said, there still exists those set of small-to-medium businesses that have had no need or requirement for a vendor enterprise solution, nor the resources to build one. Increasingly, these smaller partners are being asked to provide their services online and to have these applications support F-SSO standards. These requirements are often driven by the enterprise aiming to reduce the overhead of managing identities, reducing manual business processing and in some cases, reducing their dependency on, and exposure to risk due to nonstandard custom SSO protocols. Normally this is in line with their approach to embracing a Service Oriented Architecture strategy.

By not conforming to these requirements, the business partner risks losing a current revenue stream. In addition, they have the opportunity to gain a competitive advantage over competitors as well as the opportunity for cost savings from reduced identity management overhead. The Federated Identity Management Business Gateway addresses the SMB requirement, providing a

lightweight implementation of the F-SSO standards, without the dependency on an enterprise security product such as Tivoli Access Manager. Having the Federated Identity Manager Business Gateway deployed can also provide the SMB customer an opportunity to act as an identity provider to other partners.

Note: It is not a requirement that the partner to a Tivoli Federated Identity Manager Business Gateway deployment be Tivoli Federated Identity Manager, or vice versa. Any vendor who implemented to the supported F-SSO standards can participate in a Tivoli Federated Identity Manager federation.

The SMB pattern models the relationship between an enterprise deployment of Tivoli Federated Identity Manager with a deployment of a Tivoli Federated Identity Manager Business Gateway. The following sections outline Federated Identity Manager Business Gateway acting as the identity provider as well as the service provider.

At the time of writing, Federated Identity Manager Business Gateway supported the SAML 1.0/1.1 protocol.

Federated Identity Manager Business Gateway SP

In this architecture, the Federated Identity Manager Business Gateway acts as the service provider.

The identity provider deployment might be represented by a high availability deployment such as Figure 26-8 on page 816. The protocol flows and requirements for configuration are typical of any identity provider deployment outlined in 26.1.2, “Base pattern” on page 807.

The following sections outline the configuration of Federated Identity Manager Business Gateway as the service provider.

IIS point of contact at service provider

As with any service provider deployment, access to the *assertion consumer service* endpoint for unauthenticated users is required. Figure 26-9 on page 821 shows an IIS deployment as the service provider.

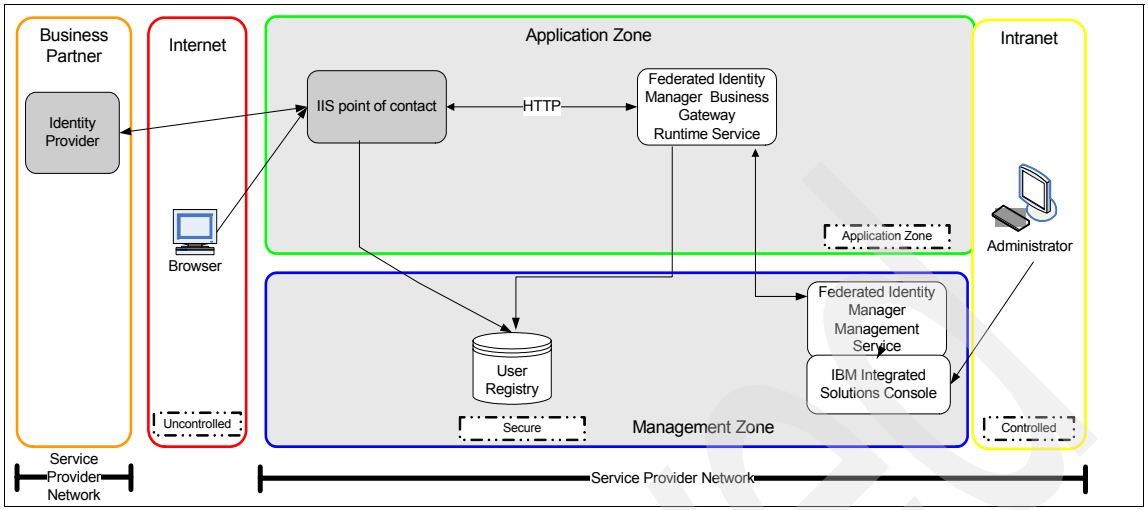


Figure 26-9 Tivoli Federated Identity Manager Business Gateway with IIS as service provider

On an incoming request containing an artifact (browser artifact profile) or assertion (browser post profile) the following applies:

- The IIS plugin sends an HTTP request to the Federated Identity Manager Runtime service located on the WebSphere instance. This represents the assertion consumer endpoint.
  - If configured for the browser artifact, the Runtime Service sends a request for the assertion to the IdP via SOAP and extracts the assertion from the response.
  - If the browser posts, it extracts the assertion from the POST body as passed by IIS.
- The Runtime Service performs token validation, mapping, and exchange.
- On receiving the response from the Runtime Service, the IIS plugin inspects the response for the presence of a user header that symbolizes the authenticated user. This header information is then used to create an IIS login context for the user.
- The IIS plugin then redirects the user to the requested application, adding any configured headers to the request.
- Ultimately the users' browser is returned both the WebSphere LTPA token from the Runtime Service as well as any application session cookies used by the IIS application.
- The user can continue to access the content provided by the IIS server or .NET application.

Again, no dependency on Access Manager exists here. Obviously some sharing (either mapping or same use of) of directory infrastructure is required in order to ensure that credential information is consistent between the Runtime Service and the IIS infrastructure.

WebSphere Application Server at service provider

As with any service provider deployment, access to the *assertion consumer endpoint* for unauthenticated users is required. Figure 26-10 shows a WebSphere Application Server deployed as the point of contact at the service provider.

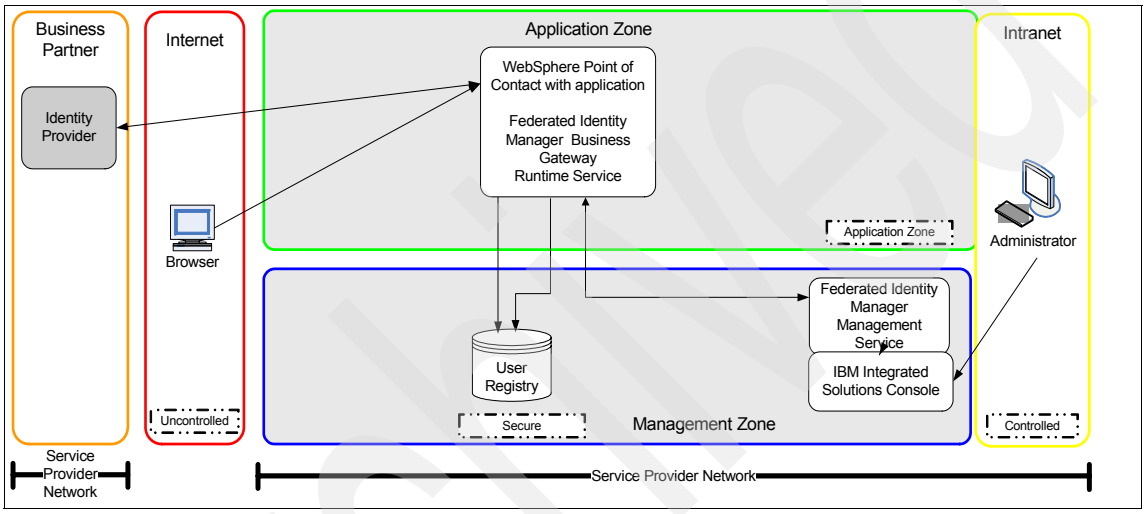


Figure 26-10 Tivoli Federated Identity Manager Business Gateway with WebSphere as service provider

On an incoming request containing an artifact (browser artifact profile) or assertion (browser post profile) the following applies:

- It is passed to the Runtime Service, which takes the incoming request:
  - If configured for browser artifact, it sends a request for the assertion to the IdP via SOAP, and extracts the assertion from the response.
  - If the browser post, it extracts the assertion from the POST body.
- Validates the assertion, maps it to a local identity if required, and returns an identity to the point of contact.
- The point of contact then creates an authenticated context for the resulting user using a JAAS login module.
- WebSphere Application Server then returns an LTPA token to the user for session maintenance.

- The user continues to access the content.

Again, no dependency on Access Manager exists.

Federated Identity Manager Business Gateway IdP

The alternative to the above configuration is the hosting of the identity provider at the SMB customer. This scenario is less likely to occur, but is relevant none the less. In this configuration, WebSphere Application Server acts as the point of contact at the identity provider shown in Figure 26-11.

Note: IIS is not supported as the point of contact at the identity provider in a Federated Identity Manager Business Context deployment.

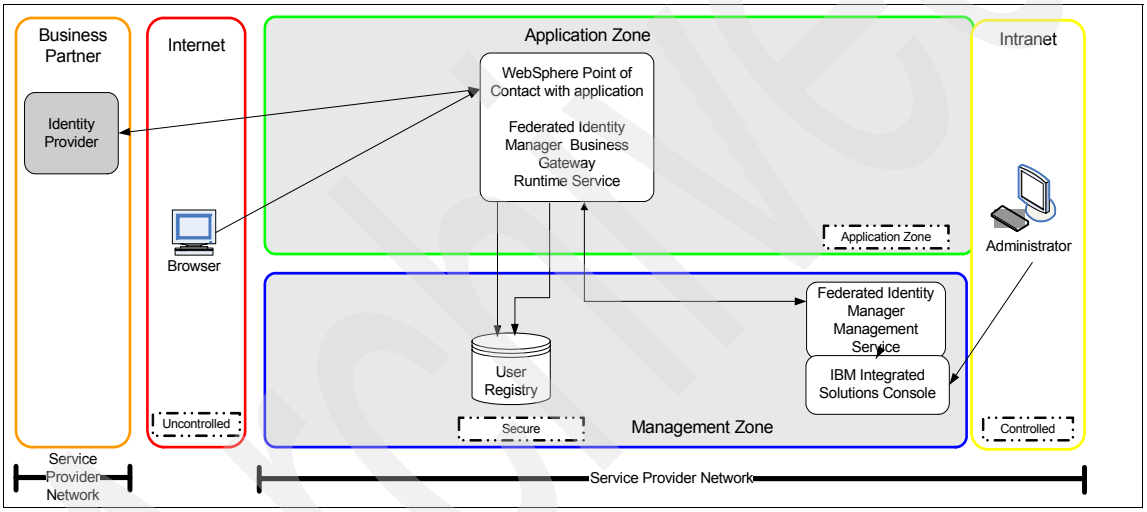


Figure 26-11 Tivoli Federated Identity Manager Business Gateway with WebSphere as an identity provider

Within an identity provider configuration, WebSphere security protects access to the single sign-on service. As such, a user must be authenticated by the WebSphere Application Server in order to access the Federated Identity Manager Runtime Service. The authentication module can be configured according to the customer's requirements, and the resultant WebSphere identity is passed to the Federated Identity Manager Runtime Service for partner token creation.

26.2 Federated Web services architecture patterns

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

Technically, Federated Identity Manager provides token validation, issuance (and exchange), identity mapping, and request authorization within a secure Web services environment. Federated Identity Manager therefore supports scenarios such as those requiring the *normalization* of the security policy applied to a Web service. In this type of scenario, an application is deployed as a Web service with one security policy (user must have role of *manager*, or the incoming request must include a SAML assertion), even though not all requestors can satisfy this policy. Federated Identity Manager can be used to provide the identity and attribute mapping required to determine the user's local roles based on those asserted by the requestor, so that the request includes the appropriate role of *manager* instead of *Partner_Manager*, for example. Similarly, Federated Identity Manager can be used to provide token exchange functionality, so that a trust partner coming in over a VPN with a UsernameToken in the <Security> header can have their request normalized to include the required SAML assertion without requiring the partner to expand their capabilities to generate the required assertion.

26.2.1 Architecture approach

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

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

In addition to providing the trust service/security token service, Federated Identity Manager includes WebSphere Application Server specific components to provide the integration of WebSphere Application Server and Tivoli Federated Identity Manager. A WS-Trust client is provided to allow WebSphere Application Server (through the WS-Security functionality) to invoke the Federated Identity

Manager trust service/security token service. Federated Identity Manager also includes a JAAS login module that allows a SAML assertion to be used to create a JAAS login context in a WebSphere Application Server. These WebSphere specific components of Federated Identity Manager that are related to Web services are collectively referred to as the Web Services Security Management components of Federated Identity Manager. In Federated Identity Manager v6 Web services security management components are provided for WebSphere Application server v5.1 and v6.0. Within Tivoli Federated Identity Manager 6.1, support is provided for WebSphere Application Server to call to the WS-Trust interface for outgoing messages. This provides the ability for outgoing messages, from WebSphere Application Server, to support additional token types provided by Tivoli Federated Identity Manager, for example, SAML tokens.

Token validation and exchange

At its most simple, the Federated Identity Manager security token service provides token validation and issuance functionality. Token validation is the process by which a token, received at the STS is validated in terms of signatures on the token, expected structure and contents of the token, and decrypting of the encrypted contents (if any) of the token. Token issuance is the process by which a (new) token is created and returned to the (requesting) Trust Client by the Security Token Service. Together, token validation and issuance can be used to implement *token exchange.* Token exchange allows for the validation of an incoming token type (such as a received SAML assertion) and the issuance of a locally valid token (such as an Access Manager for e-business compatible credential, as is accomplished by the STS in a single sign-on scenario).

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

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

Web services resources may be deployed with one particular requirement on the expected incoming token type. Requestors may be able to include only a subset

of possible token types in a Web services request. The Tivoli Federated Identity Manager TS/STS may be used to bridge this token type gap between requestors and resources.

Identity mapping

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

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

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

Authorization

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

A similar model can be applied to Federated Web services, with coarse-grained authorization performed at the boundary for incoming Web service requests. In this case, an XML gateway provides the authorization enforcement point, but Access Manager for e-business (via Tivoli Federated Identity Manager) can still be used as the authorization decision point. Tivoli Federated Identity Manager can optionally perform an Access Manager for e-business authorization API call

to determine of the requesting user is authorized to access the service being requested. Since this implemented in the Tivoli Federated Identity Manager Trust Service, the Access Manager for e-business call is transparent to the XML gateway and the requestor/provider applications. Any authorization failures result in the Web service request being rejected at the gateway and a SOAP fault returned to the requestor.

26.2.2 Point-to-point pattern

This pattern is included here for completeness, but it is not envisaged that this pattern will be used in many situations with Federated Identity Manager v6.

Tivoli Federated Identity Manager v6 does not include Web services security management support for outbound Web service requests from the WebSphere Application Server or WebSphere Application Client containers. So any token creation required at the client side of a Web services request would either need to be directly supported by the WebSphere Application container (which does not currently support SAML assertion tokens) or the Web service requestor would need to directly invoke the Tivoli Federated Identity Manager trust service to create the required token. The Federated Identity Manager trust service provides a SOAP-based interface that implements the WS-Trust (draft) standard.

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

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

26.2.3 XML gateway pattern

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

Web services requestor

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

The Web services requestor side of the XML Gateway pattern for Federated Web services can be illustrated as shown in Figure 26-12 on page 829.

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

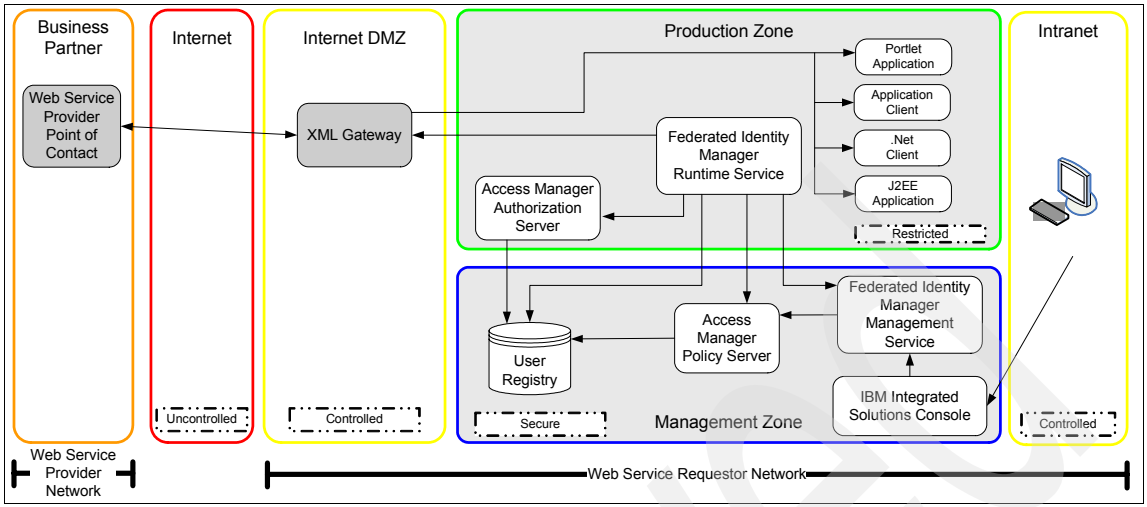


Figure 26-12 XML Gateway pattern for Web service requestor

Web services provider

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

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

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

Figure 26-13 on page 830 illustrates the XML gateway pattern for the Web service provider.

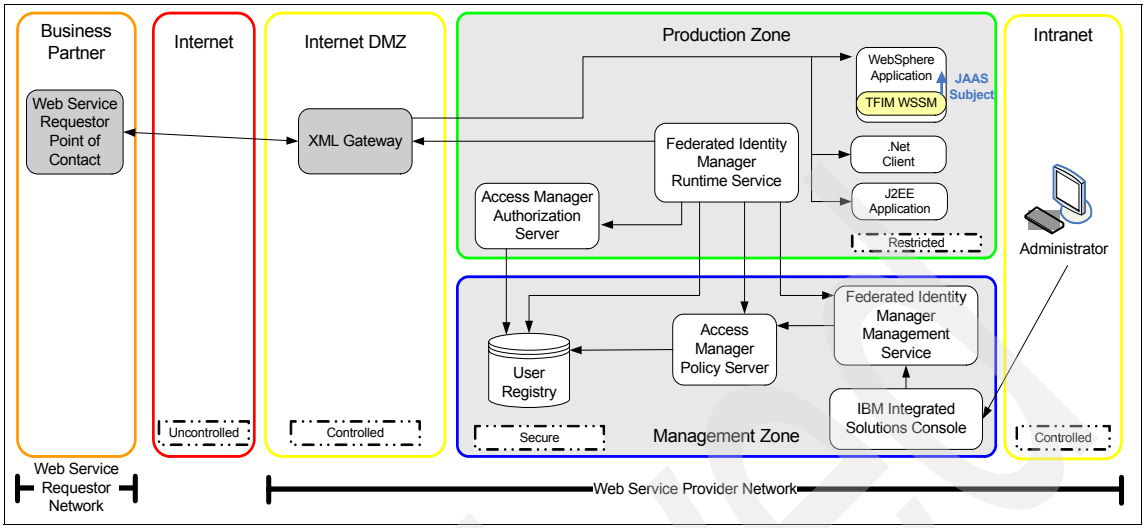


Figure 26-13 XML Gateway pattern for Web service provider

This pattern can be extended to include Access Manager for e-business WebSEAL in front of the XML gateway, with WebSEAL in the DMZ and the gateway moved inside the domain firewall. Motivation for doing this may include a desire to move the XML gateway from an outer DMZ to an inner DMZ or even into the protected segment of the network. This allows point-to-point security to Access Manager for e-business, so that Access Manager for e-business can exclude any incoming requests that do not pass simple transport layer security requirements. This provides an extra layer of protection for the keys used to encrypt or decrypt and sign or validate messages while also providing an edge-level security layer.

Supported Gateways

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

Tivoli Federated Identity Manager ships with several Web services security management (WSSM) components that enable the IBM WebSphere Web services Gateway v6 to use the Tivoli Federated Identity Manager Trust Service in a manner consistent with this design pattern.

As mentioned earlier in this book (see 24.4.3, “Web services gateway” on page 771), the IBM WebSphere DataPower XML Security Gateway XS40 is the most commonly used Web services gateway solution.

26.3 F-SSO application integration

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

26.3.1 Attribute flow between providers

Federated Identity Manager provides federated SSO to Access Manager for e-business, which in turn is responsible for providing SSO to applications. Access Manager for e-business may provide direct SSO to an application (or possibly the middleware on which it runs). As part of this enterprise SSO solution, Access Manager for e-business may pass data via HTTP headers back to an application. When Tivoli Federated Identity Manager is integrated with an Access Manager for e-business solution, it becomes possible for the two products to increase the scope of attribute flow, from point of contact to back end, to between partners to point of contact to back end.

As part of the integration of Tivoli Federated Identity Manager (and Access Manager for e-business) into an identity provider's environment, we must determine which (if any) attributes are to be provided to a service provider as part of an F-SSO solution. Figure 26-14 on page 832 illustrates the flow of attribute data from an identity provider implemented using Federated Identity Manager.

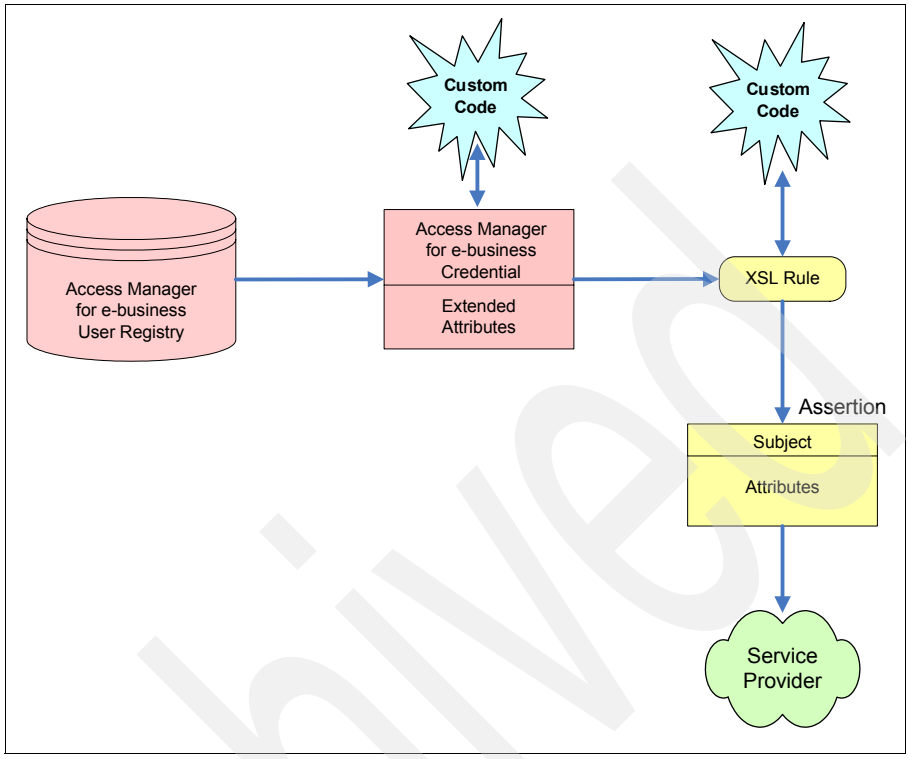


Figure 26-14 Attribute flow for identity provider

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

After the Access Manager for e-business credential is created, Tivoli Federated Identity Manager uses an XML version of the Access Manager for e-business credential as input to the identity and attribute mapping step performed as part of the assertion generation. This mapping is defined by an XSL rule. This mapping may include a simple copy of the existing (credential defined) attributes, a mapping of attributes from one value to another, or the retrieval of additional

attributes. Custom Java modules can be invoked from these XSL rules to obtain additional attribute data that is not available in the Access Manager for e-business credential. These XSL rules and any associated Java modules are invoked for every assertion generated by the identity provider and are specific to the identity provider-service provider relationship.

On the service provider side, the flow of attribute data from an incoming assertion to a service provider application is illustrated in Figure 26-15.

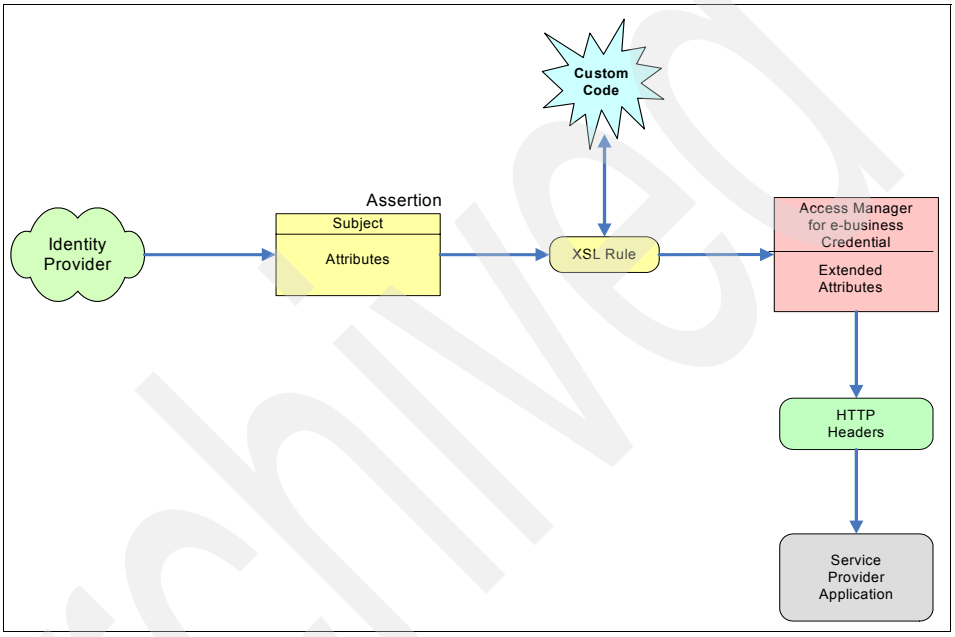


Figure 26-15 Attribute flow for service provider

In this scenario, the single sign-on assertion received at the service provider may contain attributes about a user. These attributes (and the information contained in the assertion) are translated into a Tivoli Federated Identity Manager internal format and an XSL rule is used to map this information and then format it as an Access Manager for e-business credential. This mapping may include a simple copy of the existing (assertion defined) attributes, a mapping of attributes from one value to another, or the retrieval of additional attributes. Custom Java modules can be invoked from these XSL rules to obtain additional attribute data that is not available in single sign-on assertion. These XSL rules and any associated Java modules are invoked for every assertion received at the service provider and are specific to the identity provider-service provider relationship.

Access Manager for e-business WebSEAL can then be configured to extract particular attributes from the Access Manager for e-business credential and send

the attribute values to the service provider applications via HTTP header variables. Access Manager for e-business WebSEAL allows different attributes to be sent to different applications.

26.3.2 User controlled federated lifecycle management

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

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

26.3.3 Customized user-managed federation management

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

Thus a user can be provided with a listing of *Partners you have federated with (single sign-on partners)* and a separate listing of *Partners you have not federated with*. A related example is shown in Figure 26-16 on page 835.

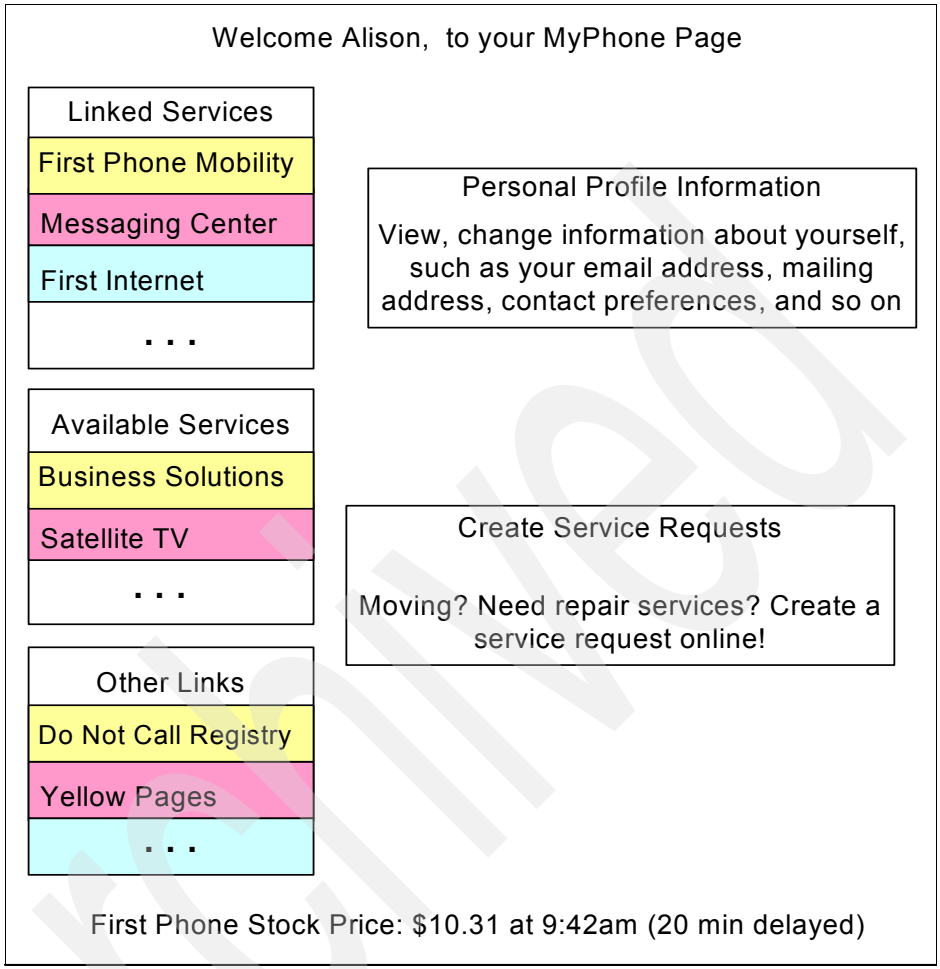


Figure 26-16 Linked services for user Alison

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

26.4 Customizing F-SSO

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

26.4.1 Customizing page templates

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

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

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

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

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

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

26.4.2 Customizing Access Manager page templates

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

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

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

26.4.3 Storing aliases

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

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

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

26.5 Solution design considerations

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

26.5.1 Exchanging metadata with your partners

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

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

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

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

26.5.2 Availability of IBM Access Manager Policy Server

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

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

26.5.3 Key management

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

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

26.5.4 Session time out

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

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

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

One solution to this requirement that will work, regardless of which vendor's products are used at the identity provider and service provider, is to have an (possibly hidden) image from the identity provider site on every service provider application page. This image may possibly be incorporated into the page design to highlight the source of the authentication. Alternatively, a servlet filter may be added to the service provider application(s) to add a hidden image to each page returned to the browser.

26.5.5 Application logout

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

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

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

A similar technique can be used in a Tivoli Federated Identity Manager environment for HTTP based SLO profiles. Javascript to delete cookies for back-end servers can be added to SLO success (and failure) page templates used by Tivoli Federated Identity Manager. However, the Liberty ID-FF standards include SOAP based profiles for SLO. With these SOAP-based profiles, the partner nodes do not have an opportunity to run any JavaScript on

the browser to delete the application cookies. It is therefore recommended that in a Tivoli Federated Identity Manager environment using SOAP-based SLO profiles, the Access Manager for e-business login page also be updated to include some JavaScript code to delete the back-end application cookies.

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

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

26.6 Conclusion

This chapter described architecture options for deploying Tivoli Federated Identity Manager and approaches for integrating this software product with other middleware and customer applications. Architecture patterns for Federated SSO and for Tivoli Federated Identity Manager Federated Web services were introduced. Then we showed you how to integrate applications into a Tivoli Federated Identity Manager F-SSO environment, and how to customize Tivoli Federated Identity Manager for F-SSO. We completed the chapter with a series of short discussions on topics relating to designing a solution for deploying Tivoli Federated Identity Manager in a real-world environment.

Part 5

Managing security audit and compliance

In Part 5 we discuss the solution that IBM offers in the security audit management space of the overall security architecture. Audit information, which generally revolves around managing intrusion and fraud, is mainly handled by IBM Tivoli Security Operations Manager. Security Operations Manager handles a multitude of integration aspects with all types of IT infrastructures and intrusion detection devices and services, which are detailed throughout this part. Audit information, which is concerned with legal and regulatory compliance, is handled by IBM Tivoli Security Compliance Manager. In addition to these specifically focused products, IBM Tivoli used the Common Auditing and Reporting Services infrastructure to target an enterprise wide centralized audit infrastructure.

27

Introducing IBM Tivoli Common Auditing and Reporting Service

Compliance is measuring how well you meet a set of security requirements. Combining disparate information to measure compliance is a difficult task. Organizations face a series of hurdles to normalize their compliance data from multiple vendors and even from multiple products from one vendor. The IBM Tivoli Common Auditing and Reporting Service presents a significant leap to measuring compliance.

27.1 Business context for compliance

Compliance measures how well a set of security requirements is met. Compliance is a large issue in light of increasing regulations designed to set an expected level of responsible behavior by companies. These regulations put a burden of proof on companies in several areas:

- Provide an effective plan to achieve compliance.
- Verify the implementation is in place and being used as designed.
- Identify areas of compliance and non-compliance.
- Show what corrective actions are taken for non-compliance.
- Identify gaps in the solution and adjust to close them.

These relate to the five steps used to design a security policy discussed in 1.5, “Security policies” on page 11. Measuring compliance becomes a more formalized approach to the ongoing cycle of security policy development and maintenance.

How do we show compliance at a practical level? We must be able to provide tangible evidence the policy has been met. The current regulatory push for compliance to a variety of new corporate responsibility laws and regulations is prompting an increased amount of formal audits from internal and external audit groups at most organizations. This audit activity is difficult due to the very nature of information technology.

Let us review some key points from our discussion about the MASS developed security and audit subsystem in 2.1.2, “MASS security subsystems” on page 21.

The purpose of this subsystem is to provide proof of compliance to the security policy. A security audit subsystem is responsible for capturing, analyzing, reporting, archiving, and retrieving records of events and conditions within an IT solution. Security audit analysis and reporting can include real-time review, as implemented in intrusion detection components, or after-the-fact review, as associated with forensic analysis in defense of repudiation claims. The security audit subsystem provides:

- Collection of security audit data, including capture of the appropriate data, trusted transfer of audit data, and synchronization of chronologies.
- Protection of security audit data, including use of time stamps, signing events, and storage integrity to prevent loss of data.
- Analysis of security audit data, including review, anomaly detection, violation analysis, and attack analysis using simple heuristics or complex heuristics.
- Alarms for loss thresholds, warning conditions, and critical events.

This becomes an incredibly difficult task. For example, regulatory agency auditors arrive at your company and announce an audit for compliance to regulation XYZ. Their audit announcement letter provides a lengthy list of information required from IT systems (including security) to measure the level of compliance. It also provides a short timeline to provide the information. The result to the business is a major resource drain to support the audit, with potential impact to projects and normal operations. While viewed as a cost of doing business, it is still a cost that needs to be reduced. Even if the requested information can be provided quickly, how do we thread through a variety of differently formatted logs and reports to show we have achieved compliance with regulation XYZ?

The key is our security audit subsystem must provide for an on demand view of compliance and audit readiness. It must be able to quickly and efficiently provide information needed to measure compliance. A side benefit of this continual posture is not only reduced resource requirements and minimized business disruption for compliance audit activity, but also a faster response to noncompliance that could result in a security or privacy breech.This awareness shows better control of IT resources and their underlying data assets.

27.2 Common Auditing and Reporting Services

Now that we have set a business context for compliance, we move to discuss the Common Auditing and Reporting Service and how it helps in proving compliance. Common Auditing and Reporting Service is the result of efforts to unify IBM Tivoli product logging and reporting. It is shipped with Tivoli Access Manager for e-business v6.0. 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, health of security servers, and so on).
- Business-related critical activities (bank transactions, insurance claims processing, order processing, and so on).
- Critical activities related to content management (updates and deletions of critical documents).
- Change Management (changes made by administrators).

Common Auditing and Reporting Service reporting is used for:

- External controls: to demonstrate compliance for various standards and legal requirements (see 1.6.2, “Legal and regulatory concerns” on page 16).
- Internal Controls: to show compliance to an organization’s security policies, as shown in Figure 27-3 on page 856.
- Checking enforcement and effectiveness of IT controls, for accountability, and vulnerability/risk analysis.
- Forensic investigations of security incidents.

Common Auditing and Reporting Service auditing is discussed in 27.2.1, “Auditing” on page 849. Common Auditing and Reporting Service reporting is discussed in 27.2.4, “Reporting” on page 852.

The Common Auditing and Reporting Service (CARS) is based on open standards and protocols.

Common Auditing and Reporting Service is shipped as a component in IBM Tivoli Access Manager for e-business Version 6.0.

The Tivoli Common Auditing and Reporting Service architecture consists of:

- Common Auditing and Reporting Service server, which includes the event server feature and the operational reports feature.
- Common Auditing and Reporting Service client, which includes the Java and C client.

Figure 27-1 on page 849 shows the Common Auditing and Reporting Service architecture.

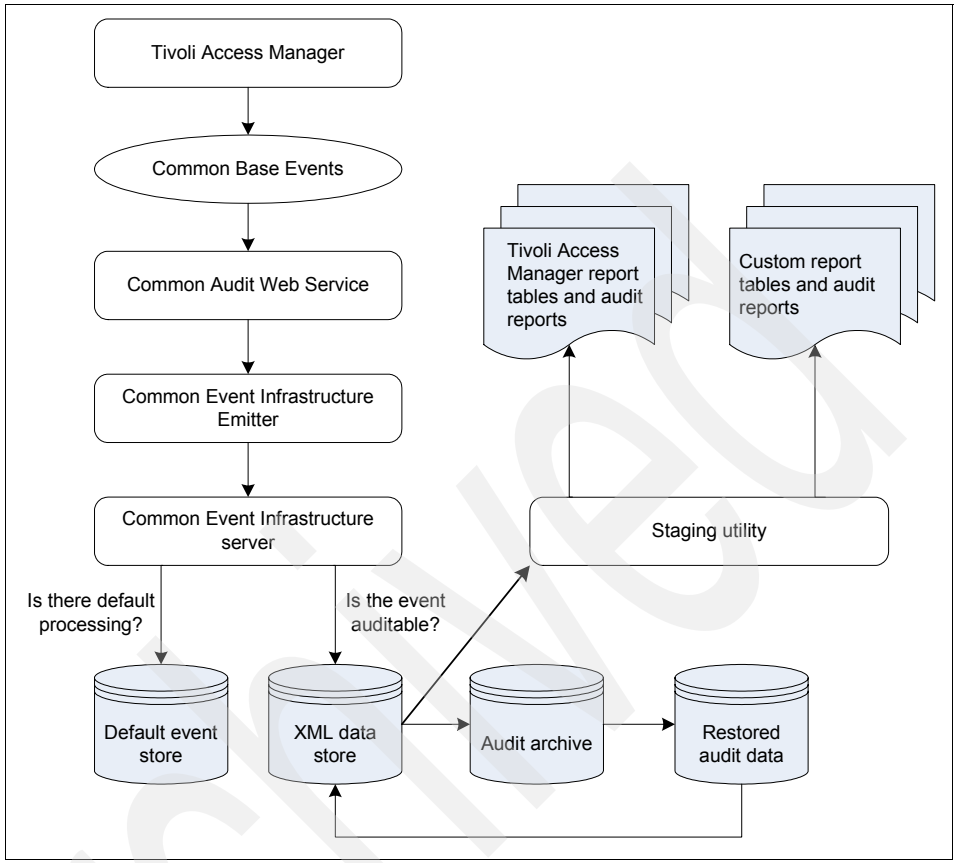


Figure 27-1 Common Auditing and Reporting Service architecture

27.2.1 Auditing

Auditing 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
- Unauthorized access to protected resources
- Modification of security policy
- Non-compliance with a specified security policy
- Health of security servers

Examples of business-related critical activities that could be audited are:

- Bank transactions
- Insurance claims processing
- Order processing

Examples of critical activities related to content management are updates and deletions of critical documents.

27.2.2 Audit logs

IT organizations can use information contained in audit logs to help them show compliance with government regulations. For this reasons, such audit logs must be maintained sometimes for years.

Audit logs are useful to check enforcement and effectiveness of IT controls, for accountability, and vulnerability and risk analysis. IT organizations can also use auditing of security-related critical activities to aid in forensic investigations of security incidents.

When a security incident occurs, the audit logs enable analysis of the history of activities (who did what, when, where, and how) that occurred prior to the security incident, so appropriate corrective actions can be taken.

For the above purposes, audit logs need to be archived (stored) and accessible for report or query for years.

Also, audit logs are typically made available in relational databases so they can be easily queried to generate reports. Facilities, such as IBM DB2 Alphablox and Crystal Reports from Business Objects, can then be used. Audit reports allow detailed review of audit data to help determine the cause of the security incident.

Based on how the audit data is used, as previously discussed, management of the audit data has the following requirements:

- Collect and store large volumes of data for a long period of time.
- Stage the data periodically (daily or weekly) into report tables for audit reports.
- Archive the audit data for a long period of time (months or years) with archival scheduled on a regular basis.
- Produce audit reports on recent and archived audit data. Such reports can be produced by customers using their reporting tool of choice or shipped as part of IBM products.

- Provide a process that is tamper resistant. That is, the audit data must be kept safe when it is generated, during the transit, and when it is stored.
- Provide auditing functionality for changes to the configuration and policy for collecting audit data.

27.2.3 Audit infrastructure

An *audit infrastructure* provides the mechanisms to submit, centrally collect, and persistently store and report on audit data, and it satisfies the previously mentioned requirements to manage audit data. The IBM Tivoli Common Auditing and Reporting Service component leverages the Common Base Event and the technologies to provide an Audit Infrastructure.

The Common Base Event is a common format for events proposed by IBM and submitted to the OASIS[1](Organization for the Advancement of Structured Information Standards) organization for standardization. The purpose of the Common Base Event is to facilitate effective intercommunication among disparate components within an enterprise. In order to effectively process audit data, it needs to be in a standard format, and the Common Auditing and Reporting Service component requires the audit data to be in the Common Base Event format.

The Common Event Infrastructure (CEI) is an IBM strategic event infrastructure for submission, persistent storage, query, and subscription of Common Base Event events. The Common Auditing and Reporting Service component uses the CEI interfaces for submission of events. Such events can be denoted as auditable using configuration options at the CEI Server in which case CEI stores them in a CEI XML Event store that meets the auditing requirements described above.

The Common Auditing and Reporting Service component allows staging of data from the CEI XML Event store into report tables. IBM products and customers can provide audit reports based on auditable events staged into such report tables. The Common Auditing and Reporting Service component also supports the lifecycle of auditable events, including archive, restore, and audit reports on restored archives. It enables common reporting against auditable events from different products and sources.

The first release of the audit infrastructure delivered by the Common Auditing and Reporting Service component is used by the IBM Tivoli Access Manager for e-business product for submitting, storing, and reporting auditable security events.

[1] For more information about OASIS, see:
http://www.oasis-open.org

Archiving and restoring audit data

The relational database schema of the CEI XML Event Store is externalized so the audit data stored in it can be archived by customers using third-party archival tools of their choice. The Common Auditing and Reporting Service component provides an XML Store utility that aids customers in archiving and restoring audit data. Also, the Common Auditing and Reporting Service supports staging of restored audit data into report tables so that audit reports can be run against this restored audit data.

Securing audit data

CEI Emitter event interfaces are protected using J2EE declarative security to ensure that only authenticated and authorized entities are allowed to use them. Transmission of the Common Base security events to the CEI Server and can be secured using SSL. Customers can protect access to the audit reports by using the access control mechanism supported by the reporting tools. Customers also need to protect the CARS XML Event Store and the report tables using the access mechanisms provided by the database.

27.2.4 Reporting

The operational reports feature of the Common Auditing and Reporting Service provides a number of compiled reports that provide information about security-related activities that occur on your system.

The compiled Crystal Reports provided with Common Auditing and Reporting Service include audit event history, password change activity, authentication event history, authorization event history, event details, resource access, and server availability reports. The Compiled Reports format allows you to run reports without having the Crystal Reports Designer installed on the system.

The following is the list of out-of-the-box reports that are available:

- General Audit Event Details Report

  Displays all information about a single auditable event denoted by the event reference ID parameter. Typically a user will run this report after running other reports and deciding an event drill down is desired.

- General Audit Event History

  Displays the total number of auditable events for each event type during a specified time period. It also shows all events of the specified event type and product name sorted by specified sort criterion and time stamp. This report can be used for incident investigation and assuring compliance.

- Audit Event History by User

  Displays total number of events for a specified user during a specified time period. It also presents a list of all events of the specified event type and product name sorted by time stamp and grouped by session ID during the time period. The purpose of this report is to investigate activity of a particular user during a specified time period.

- Failed Authentication History

  Presents a list of all failed authentication events over the time period sorted by specified sort criteria such as by time stamp. This report can be used by an administrator to investigate security incidents.

- Failed Authorization History

  Lists all of the failed authorizations events during a specified time frame.

- Locked Account History

  Displays all of the accounts that have been locked during a specified time period.

- User Password Change History

  Displays events related to password changes done by the user themselves during a specified time period.

- Administrator and Self-Care Password Change History

  Displays events related to password changes done by the user and the administrator during a specified time period.

- Server Availability Report

  Shows the availability status of Security servers on a specific machine. The user can display all protected machines in the report or limit the report by entering a single host name as the subject of the report.

- Certificate Expiration Report

  Allows detection of soon-to-expire certificates and highlights the need to replace the certificate to insure 24/7 operability. It shows the number of clients that have server/SSL certificates that expire in ‘x’ days. It will also show a table of client hostnames, the days until their certificates expire, and the server they are configured to.

- Most Active Accessors Report

  Shows a list of users who are the most active in the system, and can lead the administrators to investigate improper use of their resources.

- General Authorization Event History

  Displays the total number of authorization events, failed authorization events, successful authorization events and unauthenticated events during the

specified time period. Additionally it shows list of all authorization events sorted by specified sort criteria (time stamp, resource or user name) during the time period. The purpose of this report is to analyze authorization event history for incident investigation and assuring compliance.

- Authorization Event History by Action

  Displays list of all authorization events that contain the specified action sorted by resource and then time stamp during the time period specified.

- General Administration Event History

  Shows the history of general management actions done over a specified time interval. The administrator can use the report to track the actions of a user for administrative events.

- User Administration Event History

  Can be used to investigate security incidents, and to track changes to users by administrators.

- Group Administration Event History

  Can be used to investigate security incidents and to track changes to groups by administrators.

- Security Server Audit Event History

  Presents a list of auditable events related to security servers that occurred during the specified time period.

- Resource Access By Accessor Report

  Shows the top resources in terms of access/authorization events during a time period for each machine name identified. The report identifies who is repeatedly accessing resources and what resource is being accessed.

- Resource Access By Resource Report

  Shows the top accessors in terms of access/authorization events during a time period for each machine name identified. The report identifies which resources are most heavily accessed and which user is accessing the resource.

27.3 Scenarios

At this point, we apply the guidelines described in the last two sections to a security incident investigation scenario and a IT control scenario.

27.3.1 Security incident investigation

The following security incident investigation scenario shows how the audit data can be used to investigate break-in security incidents.

During a security incident investigation, the IBM Tivoli Common Auditing and Reporting Service is used to quickly retrieve the necessary data from events, increasing the speed of the investigation process.

In our example, the system administrator is investigating a security incident that occurred at 02.00 hours. Instead of looking for the authentication events in Tivoli Access Manager for e-business, the administrator extracts the *Failed Authentication History Report* from the Common Auditing and Reporting Service Event Store. With this information the system administrator is able to easily locate the author of the incident and take corrective actions.

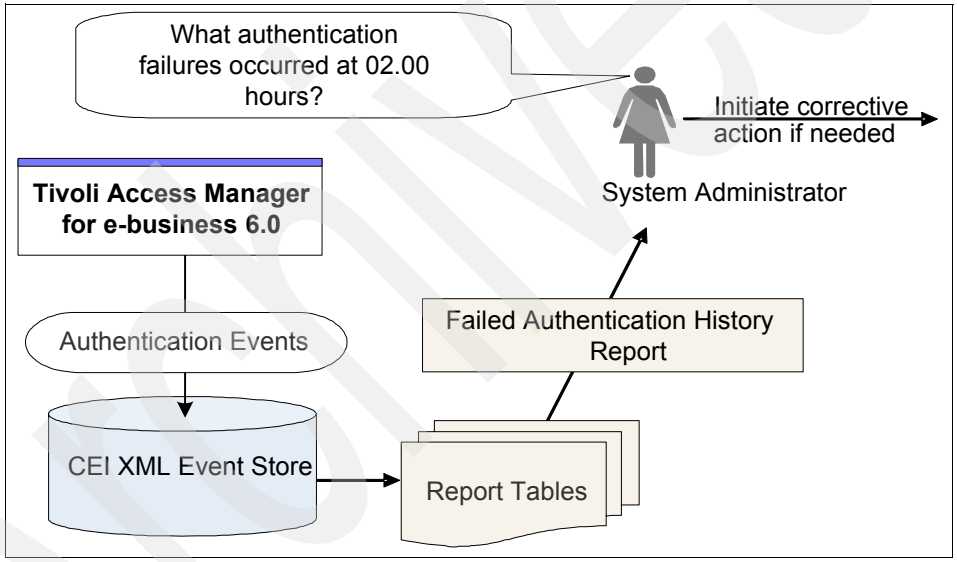


Figure 27-2 Security incident investigation

27.3.2 IT control

The following IT control scenario demonstrates how the audit data can be used to ensure that only authorized entities are accessing resources as specified by security policies to control IT.

The system administrator wants to ensure that only authorized people have access to an application or file. *The administrator extracts the General Authorization Event History* report from Common Auditing and Reporting

Service Event Store. Using this information, the system administrator can determine if there is any corrective action necessary, make the correction, and save time in the overall process.

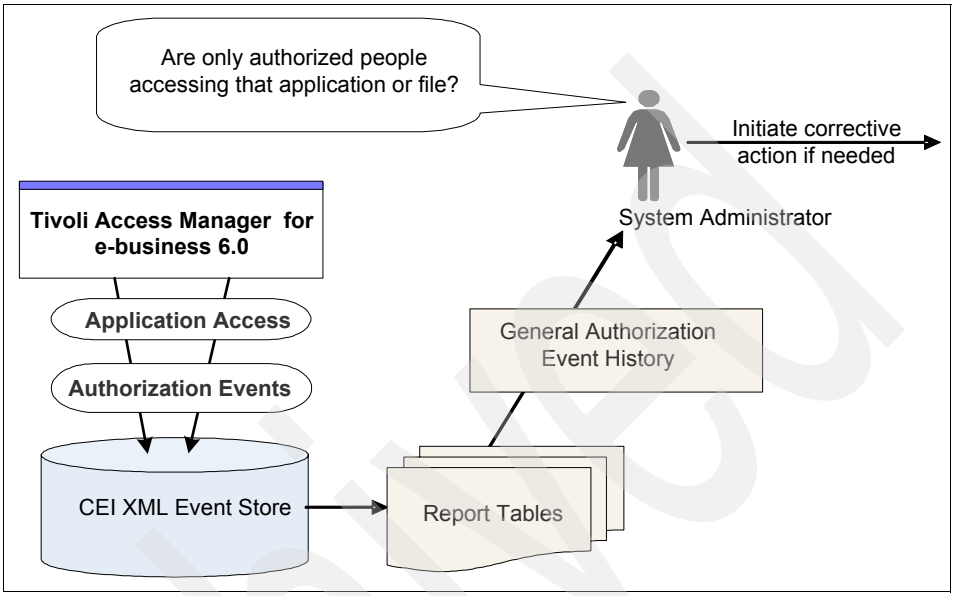


Figure 27-3 Controlling IT

27.4 Conclusion

The Common Auditing and Reporting Service provides security and audit subsytem functions that can be used to measure compliance, investigate security incidents, and verify IT controls.

28

Security Operations Manager topology and infrastructure

In this chapter we introduce the *Security Information Event Management (SIEM)* architecture. A typical enterprise security architecture that contains a myriad of different types of security related devices (for example, intrusion detection, firewalls, and network access control), as well as security logging information from enterprise applications, operating systems, databases, access and identity management infrastructure, and so on. As companies deploy more and more security related devices and applications, the need to properly manage and address information retrieved from or traversing these sources becomes more critical. To handle the ever-growing demands of analysis and correlation of security logging information, the area of Security Information Event Management was developed.

In this chapter we discuss the inherent topological architecture of IBM Tivoli Security Operations Manager, a SIEM platform designed to handle the analysis and correlation of security information from network security devices, access or identity management, and enterprise security applications.

Prior to the existence of Security Information Event Management, security administrators were forced to monitor security-related information from several sources simultaneously, such as the following:

- Network intrusion detection and prevention devices
- Firewalls
- Access and identity management systems and applications
- Enterprise application security logs

Monitoring, analysis, and correlation of information from these sources proved to be a nightmare for security administrators in enterprise security environments. Typically each vertical had its own particular management application that had to be utilized to maintain and analyze the appropriate security-related information. Administrators were forced to watch several security applications simultaneously, which made extended and detailed analysis extremely difficult in an enterprise environment. The complexity and time required to watch one particular event source, let alone cross-reference and correlate information from one event source with another event source, proved exhausting if not impossible. The industry realized the need for a new area of enterprise security architecture that was capable of resolving these key issues, and in turn the SIEM paradigm was created.

Tivoli Security Operations Manager is considered to be the central server application for analyzing security information in the Tivoli product family. Security Operations Manager manages security-related information and logs from a multitude of physical security devices and security software applications. It is considered to be the successor to Tivoli Risk Manager, and contains updated cutting-edge analysis and correlation features that were unattainable in the previous product.

Before we discuss the Security Operations Manager's logical and physical components and architecture, we take a look at the different security devices and applications that are typically found in today's enterprise IT environments.

28.1 Enterprise security devices and applications

The inherent need for an enterprise SIEM platform becomes more readily apparent as an enterprise security architecture grows and adopts a wide variety of network security devices and security applications. As noted in the previous section, managing and sifting through the vast amounts of security information generated by these event sources becomes a daunting task.

Here is the question that is usually asked first.

What products typically comprise an enterprise security architecture?

The products that are most commonly found within an enterprise security architecture deployment include the following:

- Intrusion detection and prevention systems (physical and host-based)
- Firewalls (hardware and software-based)
- Antivirus software
- Public Key Cryptography and authentication
- Access and identity management systems
- Vulnerability assessment and management

In the following sections we take a closer look at these different types of security devices and software applications.

28.1.1 Intrusion detection and prevention systems

An *intrusion detection and prevention system* (IDPS) is a type of security management system for monitoring network (*network-based intrusion detection system* or *NIDS*) and system (*host-based intrusion detection system* or *HIDS*) related security information. An IDS in general gathers and analyzes information from various areas within a network or a computer to identify possible security breaches, which include both *intrusions* (attacks from outside the organization) and *misuse* (attacks from within the organization). The IDS may be combined along with *vulnerability assessment tools* to assess the significance of an attack against the security of a host or network. An IDPS is capable of not only detecting an attack, but is further capable of stopping the attack. Such IDPS devices function by having the traffic pass through the device in a way similar to a bridge, thereby allowing the IDPS to grant only certain traffic to pass through from one physical network segment to another.

Typical intrusion detection and prevention functions for either hardware-based or software-based solutions include the following:

- Monitoring and analyzing network and system activities
- Assessing system and file integrity
- Recognizing typical patterns of attacks
- Analyzing abnormal activity patterns
- Tracking user policy violations

Intrusion detection and prevention systems have evolved greatly over the past few years, with network IDPS devices capable of analyzing network traffic at

multi-gigabit traffic speeds in real-time. An example of such a product is the *IBM ISS Proventia® G IDPS*, a network IDPS that can analyze multiple Gigabit network segments simultaneously.

28.1.2 Firewalls

A *firewall* is either a program or device, located at a network gateway that protects the resources of a private network from outside users and attacks. Security policies implemented at the network level, host level, and application level allow access only to authorized users, applications, and systems, depending on the policies defined.

An enterprise with an intranet that allows its workers access to the Internet installs a firewall to prevent outsiders from accessing its own private data resources and to control which outside resources its own users can access.

A firewall examines each network packet to determine whether to forward it towards its destination. It is often installed such that no incoming requests have direct access to private network resources.

Any abnormal attempt or traffic trying to access network resources through the firewall must be monitored actively and carefully. If there is any activity, the firewall is configured to generate an event that gets logged to an event log. This event log helps management tools, such as a SIEM platform, analyze the full extent of the event.

28.1.3 Antivirus software

Antivirus features are integrated with operating system management. This software is a class of program that searches hard drives and peripheral disks for known and potential viruses. Such software is typically capable of inoculating a virus from a host. It works to minimize the spread of infections through detection of viruses through a variety of methods. Anti-virus software utilizes signature databases as well as heuristic methods to properly identify the type and nature of a particular vulnerability like a virus or worm. Antivirus is critical to ensuring that an enterprise is free from potential threats within its environment and to minimize the damage that may be caused by a potential viral outbreak. A properly implemented antivirus detection and remediation system is a key part of an enterprise security architecture.

28.1.4 Access and identity management systems

Access and identity management systems are critical to the enterprise security architecture. Access and identity management systems utilize a combination of

directory services, authentication, and identity management to provide mechanisms for identification and authentication as well as for authorizing component access.

Access control subsystems enforce security policies by gating access to, and execution of, processes and services within a computing solution via identification, authentication, and authorization processes, along with security mechanisms that use credentials and attributes. Identity or credential subsystems generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms, the processes, and the security subsystems within a computing solution.

Access control and identity management systems provide greater control of information access within an enterprise security architecture, and they provide additional auditing capabilities that are essential to the overall security posture of the enterprise.

28.1.5 Vulnerability assessment and management applications

Vulnerability assessment and management applications are a key part of an enterprise security architecture. Vulnerability assessment is the process of analyzing, quantifying, and assessing the risk that is tied to a vulnerability or group of vulnerabilities that affect a system. While there are several methods of performing a vulnerability assessment upon a system, the result of the assessment when tied to a management application is critical to the enterprise security architecture. A comprehensive vulnerability assessment and management application utilizes vulnerability analysis to determine which areas (for example, services) are at risk to attack and to provide a recommended set of solutions to mitigate any possible risk.

Vulnerability assessment plays a crucial role when tied to other enterprise security solutions. When used in conjunction with an intrusion detection and prevention system, vulnerability assessment can provide extended information as to whether an attack, detected by the IDPS, is capable of successfully exploiting a vulnerability on a system. Vulnerability assessment and management is also critical to patch management applications, to ensure that systems are updated with the latest software versions from vendors. Newer software revisions are more likely to contain fixes to previous security vulnerabilities that were left open by earlier software versions.

An actively maintained vulnerability assessment and management program is crucial to ensuring the success of an enterprise security architecture. With an active vulnerability assessment and management program, organizations can ensure knowledge of security vulnerabilities within an organization, and what actions are necessary to remediate those vulnerabilities.

28.2 Logical components and architecture

Tivoli Security Operations Manager was architected to handle the growing demands of an enterprise security architecture. With the number of security devices and applications growing at a rapid rate, the amount of information that is generated by these devices is increasingly alarming. Without a properly designed security event gathering and analysis architecture, retrieval and storage of the information from these devices is virtually useless given the shear mass of information that is generated. Tivoli Security Operations Manager enables security administrators and analysts to analyze and manage security event information in real-time.

The Tivoli Security Operations Manager internal application architecture consists of three main components that are detailed in this section. Each of these components is critical to the operation, and each plays a major role in itself within the application. A detailed understanding of these components is critical in order to understand the internal architecture of Security Operations Manager.

Figure 28-1 on page 863 depicts an overview of the overall solution architecture. The enterprise security devices and applications, shown in the left column were addressed in section 28.1, “Enterprise security devices and applications” on page 858.

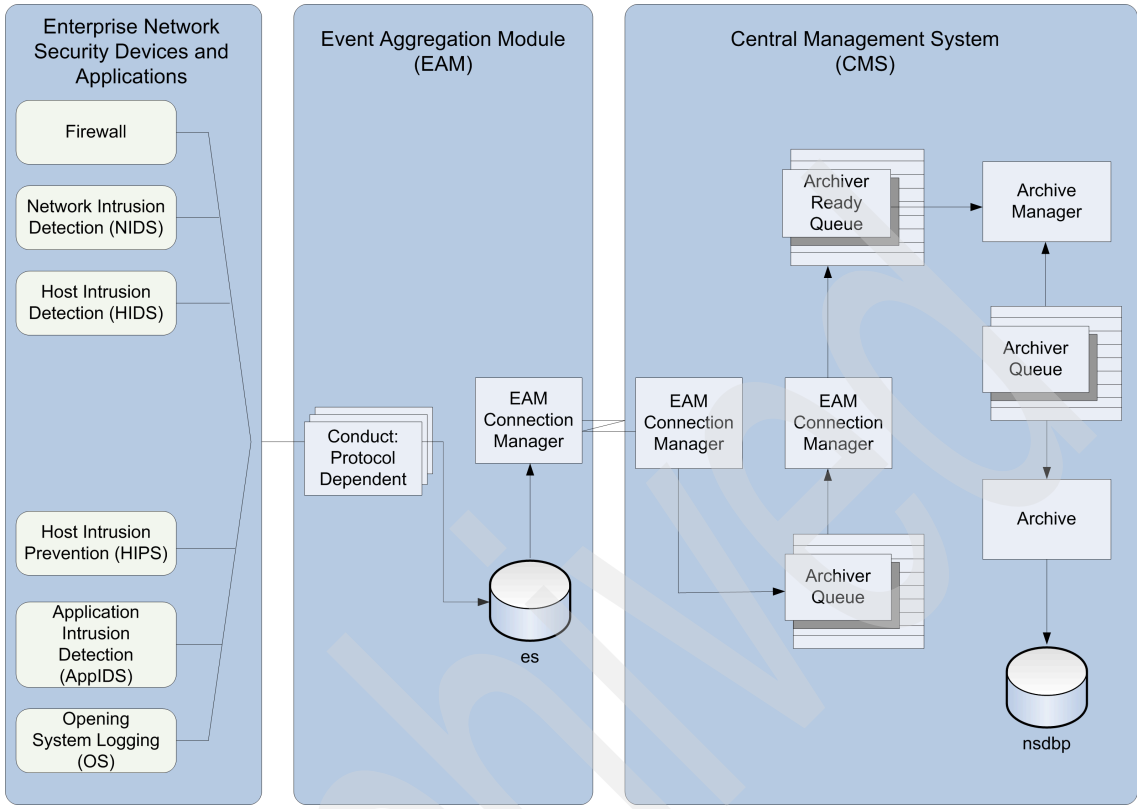


Figure 28-1 Internal Tivoli Security Operations Manager architecture

Following are the three main internal software components of Tivoli Security Operations Manager, depicted in the middle and right column in Figure 28-1:

- Event Aggregation Module (EAM)

  The *Event Aggregation Module* performs the task of gathering data from the various network security devices and applications, normalizing that data and then filtering, batching, and transmitting that data to the Central Management System (CMS).

- Central Management System (CMS)

  The *Central Management System* acts as the hub for Security Operations Manager, bringing together event data streams from all of the deployed EAMs. The CMS correlates this event data, performs *deterministic threat analysis*, calculates the threat posed to the destination by the event, and applies the rules configured in the *stateful rules engine* to the event stream, allowing the system to respond to specific attack signatures and events of

interest. Both the real-time and persistent data is used in presenting relevant information through the user interface and advanced analytical module.

- Event Archiver

  The *Event Archiver* handles security event information in queues and prepares them for writing into a persistent storage database in either DB2 or Oracle. The event stream is passed on in real-time from the CMS to the Event Archiver, ensuring no events are lost before they are written to the database.

Before we start to examine each of these components in more detail let us take a look at the processes behind Security Operations Manager and the role each of those play.

28.2.1 Processes

One of the biggest challenges security administrators and analysts face with any enterprise security architecture is finding critical threats and attacks to that infrastructure. *Without* a properly implemented SIEM platform, this challenge becomes extremely difficult due to the following reasons:

- Disparate point products that are widely distributed.
- Multi-vendor products without common formats or communications.
- Inadequate time to manually examine critical logs.
- No business-relevant context to the data.
- No inherent link between attack data and host susceptibility.
- Lack of automation.

Tivoli Security Operations Manager addresses all of these issues, acquiring disparate security event data and then applying processes required to discern the business relevant incidents in an automatic and efficient manner.

Tivoli Security Operations Manager utilizes a set of *four processes* for the collection and analysis of security event information internally in the application. These processes are actual concepts, and the end result produces the refined information the security analyst/administrator is interested in.

A visual depiction of the concepts is detailed in Figure 28-2 on page 865.

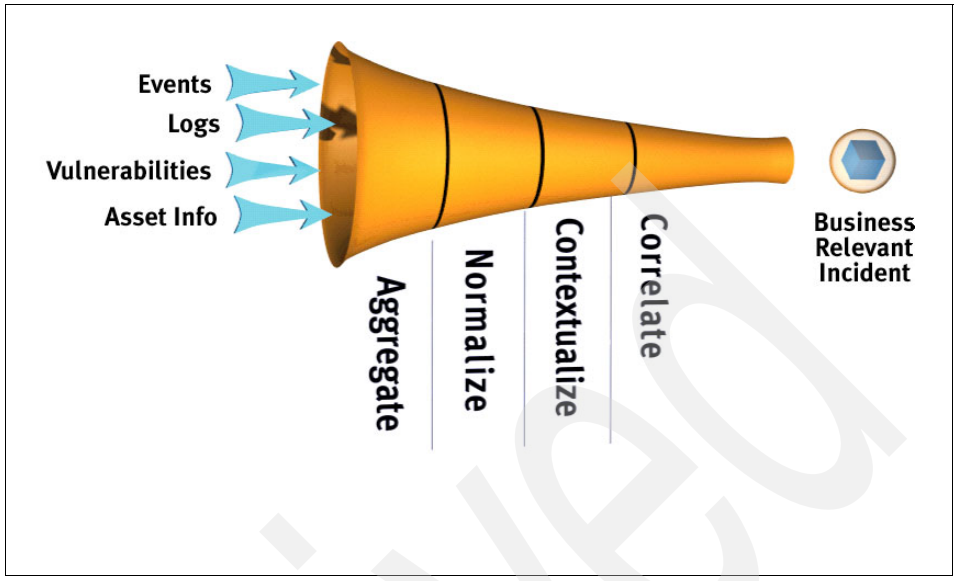


Figure 28-2 Overall view of analysis and correlation processes

Following are the four concepts internal to the application:

- Aggregation
- Normalization
- Contextualization
- Correlation

Let us take a closer look at each one of them.

Aggregation

The *aggregation* of security event information is the method of entering security information for analysis by the application. Given the administrative, logistical, and political challenges that come with deploying and maintaining agents on end devices, Tivoli Security Operations Manager provides fully featured aggregation using common protocols such as XML, Syslog, syslog-NG, SNMPv1-3, Check Point OPSEC, Cisco IDS, AVDL, and SMTP utilizing an agent-less method. Through these collection methods, Tivoli Security Operations Manager is able to provide the most comprehensive support for network oriented standards than any other security management solution.

Tivoli Security Operations Manager also utilizes a low impact, customizable agent for situations where a unique application or device contains security event information critical to the enterprise. These agents provide the flexibility needed

to acquire any data found in your enterprise, whether in a database, in a file, or in a proprietary event log format, with minimal impact to the critical system that it is monitoring.

After the data is collected we still face the challenge of effective analysis since there are no universally accepted standards for security event logging in the information security industry, and vendors are free to capture event data in any proprietary format. This creates a level of complexity in the analysis that prevents the ready distillation of information from the native data sources. This problem can be effectively mitigated by standardization of the data and its format.

Normalization

The process of *normalization* places security event information and data into a common format containing standard fields. In short, normalization enables efficient storage, processing, and retrieval of information relative to the security posture of the enterprise.

The first stage of Tivoli Security Operations Manager's two-step normalization process is to break an event into its component parts and place each of these parts into an individual database field. This step provides an accurate map between the native source data and the subsequently normalized data. The second stage transforms the device-specific event information into generic event classes. Together, these *event classes* provide a common taxonomy that can be used to analyze, search, and report on activity without prior knowledge of the specific device reporting the data. This allows for a holistic view of the enterprise's security environment.

As new device types are integrated, the devices' specific, proprietary event type conventions are automatically transcribed into the common event type conventions, allowing the correlation to work with event types across different security device brands. For example, an *accept* event from one firewall vendor and an *accept* event from a different firewall are automatically mapped to the generic *fw.accept* class. This advanced feature enables the security staff to accurately analyze data from the myriad of disparate devices throughout the enterprise.

Contextualization

After events are normalized, they are then placed into *business context*. By adding business relevance to events, resulting incidents can be listed in context of business priorities. Some of Tivoli Security Operations Manager's features that help to apply business relevance include the following:

- Security domains are logical groupings of sensors associated with permission-based profiles. For example, each business unit can have its own security domain. Therefore an incident pertaining to a particular security

domain can be weighed higher or lower depending upon the business' priorities (for example, accounting versus marketing).

- Watchlists are logical groupings of Hosts or Netblocks that are not restricted to one security domain. Typical watchlists include *Sarbanes-Oxley related Assets*, *VPN users*, or *Perimeter Devices*.
- Event taxonomies are classification systems that allow for categorization of events. This supports the ability to view all data in terms such as *User Login* and *Failed Login* in order to provide metrics for many things, especially regulatory compliance. Tivoli Security Operations Manager includes built-in detailed event mappings for approximately 15,000 log and event types from a broad variety of security products and systems.

Correlation

After business context is applied to the security-event data, *correlation* is the next necessary step. The correlation engine correlates the events against a host of factors to determine attacks and misuse.

In order to accurately and comprehensively detect threats to the enterprise, Tivoli Security Operations Manager employs a unique methodology, based on four complementary correlation techniques: Statistical Correlation, Rulesbased Correlation, Vulnerability Correlation, and Susceptibility Correlation. While most security management platforms rely primarily on Rules-based Correlation, applying multiple methods of event analysis is important because different techniques are better suited to detect different types of attacks, misuse, and policy violations.

By combining these methods, Tivoli Security Operations Manager can uncover attacks and misuse what would normally be hidden from view, using only the existing data in your security infrastructure. For example, the Statistical Correlation technology has been shown to provide superior analysis of anomalous behavior out-of-the-box. In contrast, the Rules Engine is critical for detecting policy violations. Tivoli Security Operations Manager is the only security management platform that integrates four distinct correlation techniques, offering *defense-in-depth* within a single security management platform.

The correlation process is based on both the source and destination IP addresses, ensuring the capture of all information related to the event. This extensive correlation enables Tivoli Security Operations Manager to provide a comprehensive representation of the up-to-the-minute security posture and enables the effective prioritization of threats. This in turn maximizes the effectiveness of the enterprise security staff and operations teams, allowing them to investigate the most important and relevant incidents first.

Tivoli Security Operations Manager's *four stage correlation process* enables scalable incident recognition and precise policy enforcement:

- Statistical correlation

  Tivoli Security Operations Manager's Statistical correlation engine reaches beyond basic rule sets, using patent-pending algorithms to present users with unique insight into anomalous activity on their network. It provides users with significant out-of-the-box value, as it can detect threats that bypass signatures such as new attacks that were not seen before. It can also identify unknown items, such as the source of attacks that are outside the system and present them as Source Addresses.

  Computational correlation uses embedded algorithmic logic that operates on every significant event that is monitored by the system. The technique employs a number of inputs such as event validity, event priority, and the criticality of the assets involved. It then considers statistical variables, such as event frequency, to score and prioritize the most suspicious sources of activity as well as the most threatened hosts in the network. This information is presented to users in the main dashboard. Statistical correlation is exceptionally easy to implement and use. The algorithmic logic is embedded into the product and requires minimal administration. By using embedded mathematical algorithms and tunable parameters that are default upon initial configuration, an organization can very quickly deploy the platform. Tuning over time makes the algorithms more and more effective. It is a high-performance method of analysis that enables real-time results, and it is not subject to the performance decay that Rules-based Correlation suffers over time.

  Real-life examples: Large enterprise customers estimated that statistical correlation capabilities alone identified approximately 70% of the incidents of interest. Another customer was alerted to an unknown attack by Tivoli Security Operations Manager's statistical correlation capabilities and was able to mitigate it before it had a significant effect on the infrastructure. It was later identified as the SQL Slammer worm.

- Rules-based correlation

  Rules-based correlation is used to customize Tivoli Security Operations Manager to the user's specific security environment for both incident detection and for policy monitoring. The product comes with a number of rule templates based on typical attack sequences and security best practices. Users can start with these and then build up a library of rules that reflect its security environment.

  Rules can be broad or granular, but they work best in explicit security situations where a specific A+B+C sequence is clear and consistently recognized. Rules are also used to execute automated actions within the

product such as executing a firewall configuration, shell command, or opening a trouble ticket.

Rules are particularly useful in monitoring misuse and enforcing policy. For example, Tivoli Security Operations Manager's Rules-based correlation engine can analyze user-based events from a variety of security, host, and application logs. This is particularly important with the current onslaught of federal mandates, such as Sarbanes-Oxley or Gramm-Leach-Bliley, which require that controls are in place to ensure data integrity and confidentiality.

Meta events allow users to create reusable building blocks upon which they can build more intelligence into the Rules-based Correlation Engine. For example, in order to recognize a Brute force login attempt, the user could write a rule that identified repeated failed login attempts. This rule would trigger a Meta event called Password_BruteForce. The user could then write another, more complex rule that fired whenever the Password_BruteForce Meta event was followed by a successful login or a system configuration change event.

- Vulnerability correlation

  In addition to correlating disparate event data, Tivoli Security Operations Manager also correlates attack data with vulnerability data. The end result is a direct one-to-one mapping of *exploit to vulnerability* whenever such information is available. Users import vulnerability data from their vulnerability scanner (for example, ISS' Internet Scanner®, Nessus, eEye Retina, nCircle IP360, Foundstone, QualysGuard, and SPI Dynamics WebInspect), and then the data can be associated with attacks seen from the organization's Intrusion Detection Systems (IDS). Tivoli Security Operations Manager can correlate vulnerabilities with multiple scanners across product families. Thus, Tivoli Security Operations Manager provides a vendor-agnostic vulnerability correlation solution.

  Identifying a mapped exploit-vulnerability pair enables an organization to locate compromised systems and react in a timely manner so as to reduce the impact it has on the organization. For example, teams can mitigate worm damage or can catch attackers before they do irrevocable damage or reach precious information within the organization. When a match is seen, Tivoli Security Operations Manager can respond and launch an action such as an alert, an e-mail, a Meta event, an SNMP trap, a ticket, and so on.

- Susceptibility correlation

  Despite the value that an exact *exploit to vulnerability* match provides, as variations of a given exploit begin to spread, the mappings quickly lose relevance. Additionally, there can be multiple variations on an attack against a single vulnerability and these variations are impossible to foresee and categorize.

To address this, Tivoli Security Operations Manager uses a more valuable and scalable technology that sheds light on the probability of an attack's success, called *susceptibility correlation*.

Susceptibility Correlation determines the probability of an asset's exposure using all available information about that asset, such as services running, ports open, and the operating system on the machine. This technology raises visibility of threats against susceptible hosts and reduces noise of threats against non-susceptible hosts. The first phase of susceptibility correlation is in place, determining exposure based on vulnerable services and ports on specific systems.

This real-time method of analysis bubbles up to the surface the systems that are experiencing activity that is most likely to result in a compromise and reduces the criticalness of threats against non-susceptible hosts. You can benefit by having fewer threats to investigate, so your time is focused investigating those threats most likely to impact the organization. An additional benefit is that not only can susceptibility correlation provide prioritized threat data to an incident management team, it can also provide important real time prioritization to the risk mitigation team who is trying to configure and patch systems.

Susceptibility correlation is exceptionally easy to implement and use. The logic is embedded into the product, requires minimal administration, and is an out-of-the box benefit.

28.2.2 Event Aggregation Module

The Event Aggregation Module (EAM) serves as the event aggregator of security event information from all of the devices within the enterprise security architecture. One or more EAMs can be deployed in any number of configurations necessary to handle the load of event information based on a particular security architecture's need. Logically constructed, EAMs provide a central concentration point of security event information utilizing a variety of collection methods that are detailed below.

The internal structure of the Event Aggregation Module, depicted in Figure 28-3 on page 871, is very logical and concise. EAMs utilize a series of *conduits* to collect information from disparate security devices and sources within an enterprise security architecture, and then normalize the information so it can be passed along to the Central Management System.

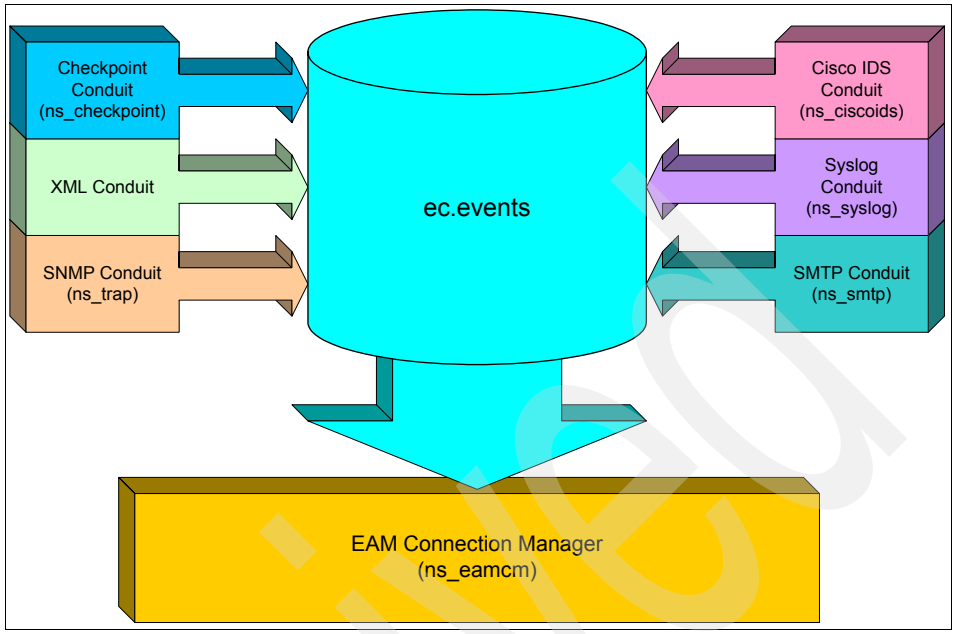


Figure 28-3 EAM internal structure

Conduits

A *conduit* is a *data collection device* designed to communicate to a group of security devices in their native protocol, normalize data from those devices, and insert that data into the memory resident database (*ec*). Conduits are key to how information is gathered and aggregated from the security devices within an enterprise security architecture.

Conduits are responsible for the following three major functions in the EAM: data collection, normalization, and data storage.

Data collection

Data collection is the key integral component of conduits in the EAM architecture. Conduits are specialized for a particular protocol (SNMP, Syslog, SMTP, and so on) in aggregating security information for normalization by the EAM. Certain conduits are utilized for a specific product or vendor, such as the Check Point and Cisco IDS conduits. The use of conduits for data collection ensures that the integration and gathering of information from the existing security architecture requires minimum configuration, while supporting the broadest range of devices.

Normalization

Normalization is the process of ensuring that disparate security event information is put into a standard, common format that is readily usable by the other components, such as the Central Management System. With different vendors supporting different event logging formats, and with different collection mechanisms utilizing different protocols, creating a normalized data format for security event information is fairly tricky. Following are the key steps to the Tivoli Security Operations Manager's normalization process within the EAM:

1. *Parse the event.*
2. Apply the *rules file* for a particular device or log type.
3. *Classify security event information* according to configurable event classes (*event_class*).
4. *Assign a priority* based on the event format (*priority*).

Let us take a closer look at these concepts.

Events that are of an unknown format or unidentifiable by the conduit are labeled internally as an *unknown event* and are passed along to the CMS for further analysis.

Enterprise security devices that utilize the XML, SMTP, SNMP, and Syslog conduits have an associated *rules file* that provides the necessary rules to identify, classify, and assign priorities to each event generated by that device. These rules files must be maintained during operation to ensure proper classification of events. As vendors update and refine their devices, resulting in new event classifications, the rules file must also be updated to reflect these changes. The Check Point FW-1 and Cisco IDS conduits use internal rules to perform event identification, classification, and prioritization.

After an *event is classified* as a particular type based on the rules file, the event is assigned to a *class_id* that corresponds to a particular *event_class* that describes the event. Event classes have the following format:

```
<device_type> . <event_type>
```

The *device_type* field describes the device that is generating the security event information, while the particular *event_type* is defined based on the events classification within Security Operations Manager. The *class_id* is simply a unique identifier in the back end database that is associated with a particular *event_class*.

Table 28-1 is an example of configured event classes and their associated *class_ids*:

Table 28-1 Event Classes and associated Class Identifiers

class_id	***event_class***
10004	os.log.audit.success
20001	app.virus.detect
20002	app.shutdown
40002	neu.auth.config.group
40001	neu.auth.config.account
50001	fw.accept
50003	fw.reject
60002	ids.host.reset
60003	ids.host.banner
60011	honeypot.detect

After an event is properly classified, it is *assigned a priority*. This priority provides a means of weighing specific events when calculating the threat posed by those events. The default value for the priority of a given event is 50%. The current range of values is 0-100% with common values being 33%, 50%, 66%, and 100%. This value provides a direct input into the calculation of the atomic threat associated with an event. Changing the priority of an event changes the atomic threat assigned to that particular event.

Data storage in the ec database

Normalized events are stored in the *ec.events* table, where they remain before they are passed along to the CMS via the EAM Connection Manager, which reads event data and sends it encrypted to the EAM Manager on the CMS.

Syslog conduit

One of the more popularly used conduits, the syslog conduit, depicted in Figure 28-4 on page 874, provides a central collection and management point for all devices that report events using the syslog mechanism provided internally by the Unix and Linux operating systems. The syslog files on the local EAM are used as a central collection point. This requires that the devices reporting through the syslog conduit be configured to forward their logs to the syslog file on the EAM.

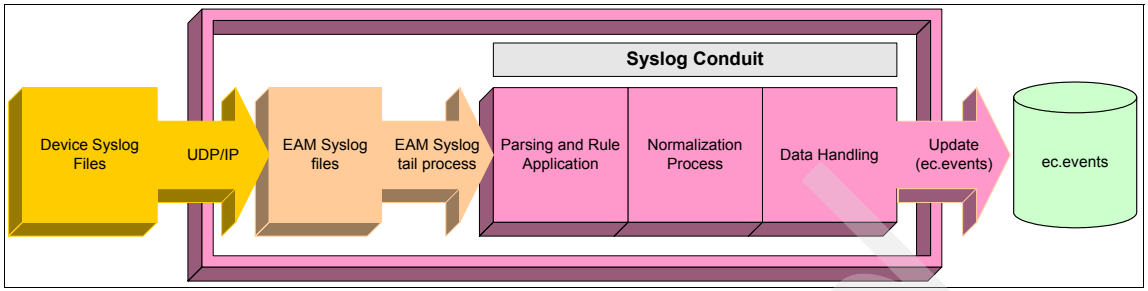


Figure 28-4 Syslog conduit

Event Aggregation Module Connection Manager

The *EAM Connection Manager* (EAM CM), as shown in Figure 28-5 on page 875, assists in managing the communications and transmission of event data from the EAM to the CMS. The EAM CM provides a number of features that are critical for the communications and operations of Tivoli Security Operations Manager between the EAM and the CMS. The EAM CM is specifically responsible for the following:

- Reading event data from the ec database.
- Performing filtering and batching of security event information.
- Responding to a heartbeat signal from the CMS.
- Transmitting event data using an encrypted TCP/IP connection to the EAM Manager on the CMS.

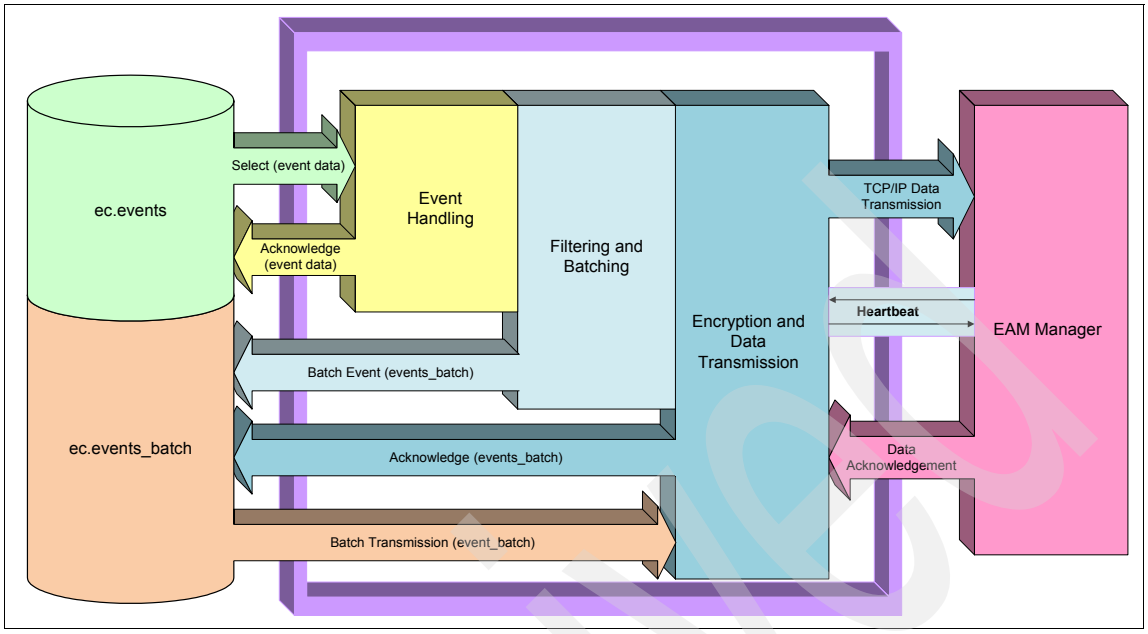


Figure 28-5 EAM Connection Manager internal architecture

Event Data Selection, Filtering, and Batching

Security event data, which is contained in the ec.events table, is selected in a first-in first-out (FIFO) manner to be sent to the EAM on the CMS. A generated SQL statement that includes any configured filters or batch information extracts the information. This information is then sent to the EAM Manager on the CMS in an encrypted format.

Filters are generated by the user on the *Event Filter Definition* window on the EAM User Interface (UI). Values entered in the Event Filter Definition are stored in the ec.filters table. A record is filtered when it matches the specific values entered in the filter definition.

Batching is accomplished by configuring the action field in a *Filter Definition* to *Batch Event*. When the event matches the filter criteria and the action is defined as Batch Event, the event is written to ec.events_batch table. The EAM CM reads from the ec.events_batch table during the period defined in the EAM System configuration and sends the events stored there to the EAM Manager on the CMS.

Data Transmission

Security event information is transmitted from the EAM to the CMS via an encrypted TCP/IP connection. The data transmission is accomplished in the following four distinct steps:

- The EAM CM first reads the data from the ec.events or ec.events_batch table, assigns an event_id, and transmits the data to the EAM Manager on the CMS.
- The EAM Manager then writes the data to the engine_queue, an operating system queue on the CMS.
- After the information is in the engine_queue, the EAM Manager transmits an acknowledgement to the EAM Connection Manager on the EAM.
- The EAM CM then updates the ec.events or ec.events_batch table by deleting the transmitted events from those tables.

This four-step process ensures the complete and secure transfer of security event information between the EAM and the CMS.

EVENT_ID

The EAM CM assigns a unique event_id to each event, allowing the event to be tracked throughout the EAM architecture just prior to transmitting the event to the EAM Manager on the CMS.

The EAM Connection Manager responds to a *heartbeat* from the CMS to provide assurance of the proper operation of the EAM during periods where data is not being passed to the CMS. This heartbeat is provided as a specialized TCP/IP packet sent by the EAM Manager on the CMS and acknowledged by the EAM Connection Manager. The heartbeat occurs on a two minute interval when data is not being sent.

Universal Collection Module

The *Universal Collection Module* (UCM), as depicted in Figure 28-6 on page 877, is an agent-based collection module that provides event data collection from file and database sources on hosts that are unable to directly communicate with the EAM or in environments where the EAM cannot be readily deployed. A typical example is when a security device management application is used as the aggregation point for security information from a particular vendor's device within an enterprise security architecture. The UCM is then deployed onto the management application server to directly aggregate the information into the EAM.

The UCM provides two methods for collecting event data:

- File Method - The file method is utilized by the UCM to collect information from those sensors that use a file structure to store event data. The UCM collects data by tailing the file where the event data is being written.

- Database Method - When the host sensor uses a relational database for event storage, the UCM utilizes a JDBC connection with the host database to extract event data as it is written to the database.

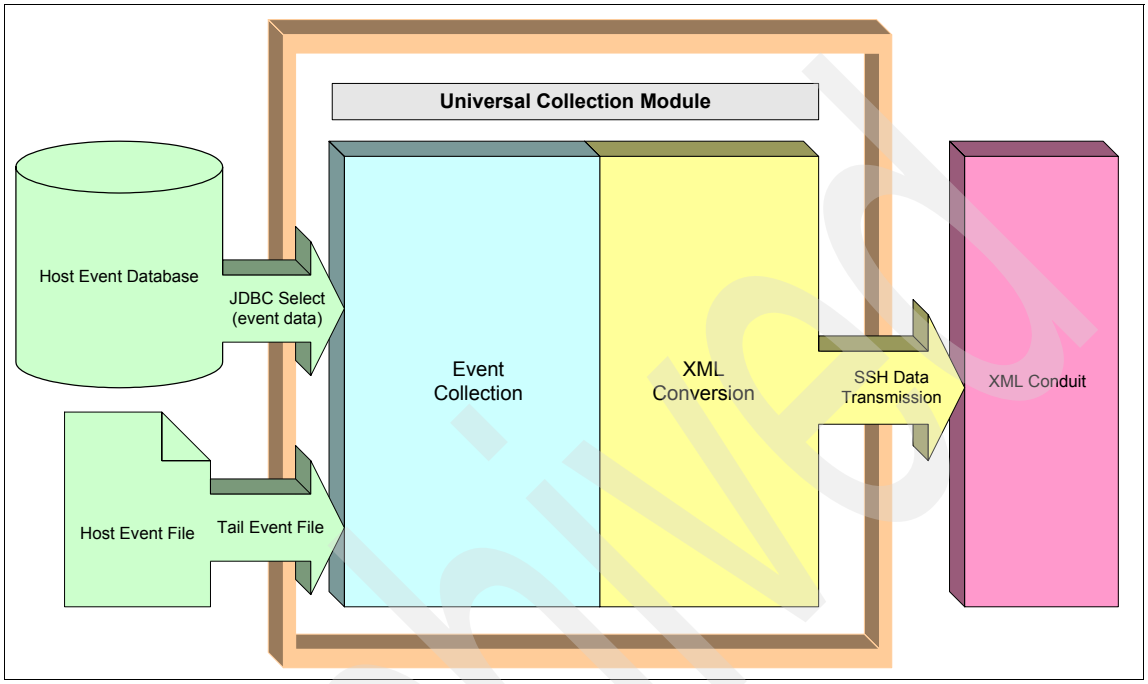


Figure 28-6 Universal Collection Module architecture

The UCM then translates and batches the event data into an XML document by wrapping the event data in XML tags defined by the UCM. The XML is then encrypted and transferred over a Secure Socket Layer (SSL) connection to the XML Conduit on the EAM.

28.2.3 Central Management System

The *Central Management System* (CMS) acts as the hub for Tivoli Security Operations Manager, bringing together event data streams from all of the deployed EAMs. The CMS, as shown in Figure 28-7 on page 878, correlates this event data, performs *deterministic threat analysis*, calculates the threat posed to the destination by the event, and applies the rules configured in the *Stateful Rules Engine* to the event stream. This allows the system to respond to specific attack signatures and events of interest.

The system directs the correlated event data stream to the archiver for persistent storage, while a running subset of the event data is directed to the Event Console

for real-time display. Both the real-time and persistent data is used in presenting relevant information through the user interface and Reporting module.

The Central Management System (CMS) provides Tivoli Security Operations Manager with a centralized data handling and storage device, capable of correlating upwards of 1000 events per second (EPS) while performing atomic and compound threat determinations, providing real-time and historic threat analysis, and taking actions on user-defined stateful rules and alert criteria. The CMS also provides a user interface with a security dashboard for monitoring events, as well as a full reporting engine.

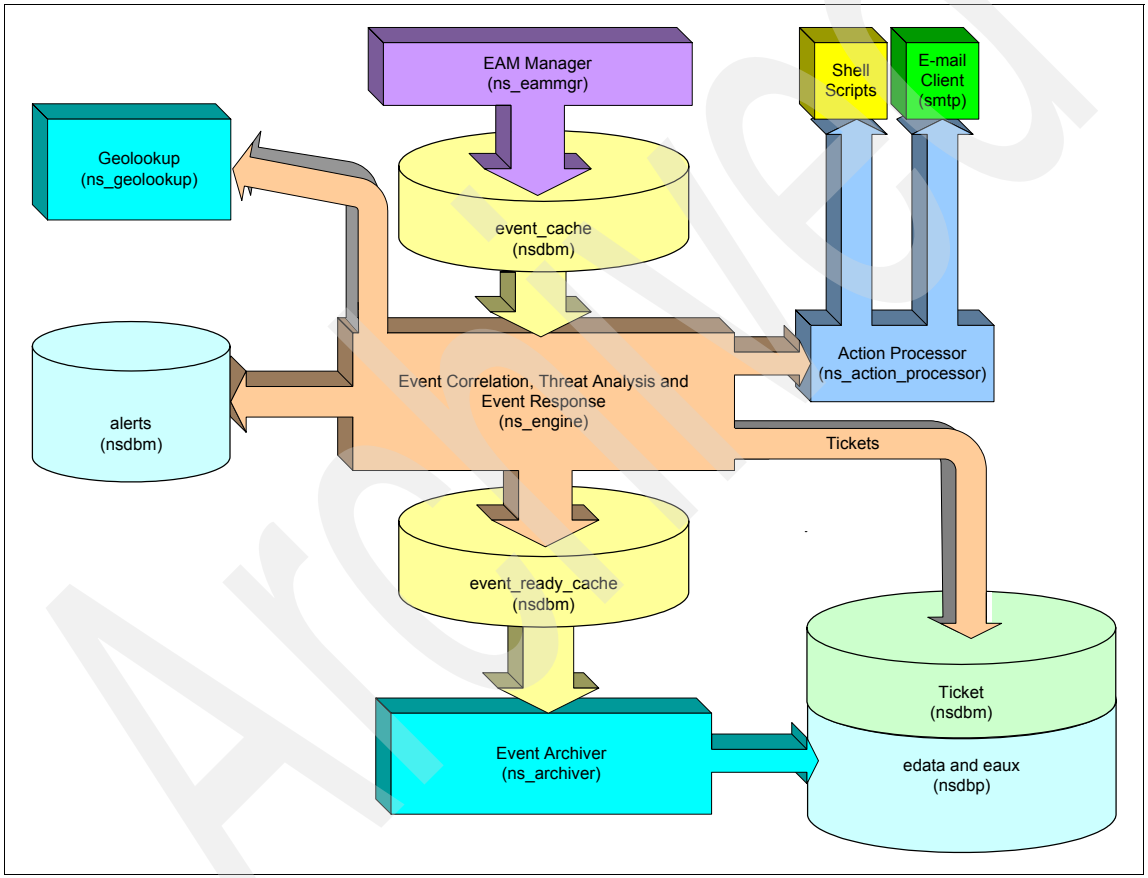


Figure 28-7 Central Management System architecture

Process overview

Events are gathered by the EAM Manager (ns_eammgr) and are stored in the event_cache. These events are then read by the Correlation Engine (ns_engine), which filters the event stream, correlates events, determines threat, performs rule

matching, passes actions to the Action Processor (ns_action_processor), creates alerts if necessary, and then writes the event to the event_ready_cache. The Event Archiver then reads the event and stores it into the database.

EAM Manager

The EAM Manager actively manages the event data stream between each of the deployed EAMs and the CMS. The data stream is managed so as to minimize data loss and maximize throughput.

Event correlation and threat determination

Event correlation and threat determination involves a programmed logic aiding in the analysis of the event data stream. This programmed logic performs many of the routine tasks currently performed by security analysts: sorting and determining the relationship between events, assigning a weighted threat value to each event, and associating each event to its source and destination hosts. Tivoli Security Operations Manager rules provide a concurrent approach to threat determination. By applying stateless and stateful rules, the CMS screens the event stream against configurable enterprise-level attack signatures, and triggers responses based on these signatures.

An example of a simple stateless rule would be an E-mail alert to be triggered when a portscan is detected by a specified sensor on any of the financial database servers within a network. An example of a stateful rule is to issue a firewall blocking command based on the scan of several ports monitored by different network security devices in conjunction with a repeated login failure indicated by syslog alerts.

Event caching and archiving

After correlation and threat determination are completed, the CMS is ready to provide relevant information to the user. The eventstream is then directed to the Event Archiver for persistent storage in the event database.

28.2.4 The Event Archiver

The Event Archiver, shown in Figure 28-8 on page 880, serves as the gateway between the Central Management System, and the primary database. As events are processed, the correlated events are written to the Archive Ready Queue, in preparation for the archival process.

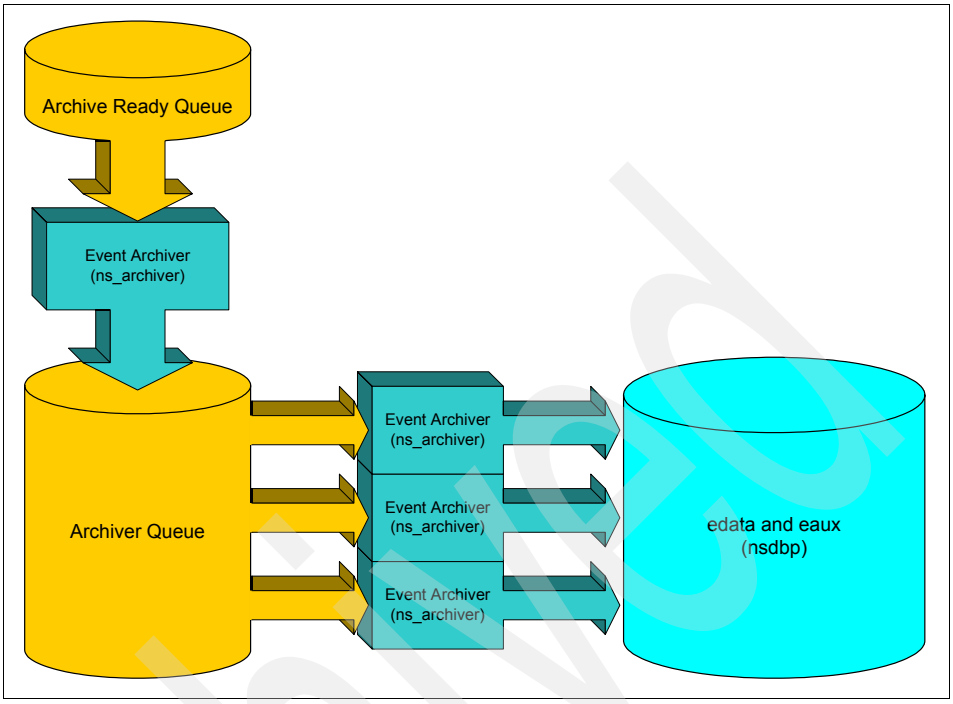


Figure 28-8 The Event Archiver

The Event Archiver reads events from the Archiver Queue and archives them in the assigned edata and eaux tables.

28.2.5 Additional logical components

To paint a complete picture of the Tivoli Security Operations Manager logical components we want to add two more that were not individually depicted in Figure 28-1 on page 863. These components are the Web interface and the reporting.

Web interface

The Web interface provides the event information. Most production networks can generate a huge number of events, which can easily overwhelm the security analyst. To alleviate this problem, the main event investigation tool, the *PowerGrid*, allows similar events to be displayed together.

The CMS and the EAM both have a Web-based interface. This interface lets administrators manage these components and view their current status. The CMS' interface also lets security analysts investigate events, configure

correlation, and generate reports. We strongly advise you to use secure socket layer (SSL) connections in order to secure the Web-based interface.

The CMS' Web-based interface has the following three purposes:

- View and investigate events in near real time, while they are happening.
- Produce reports.
- Configure Tivoli Security Operations Manager:
  - Network configuration
  - EAMs
  - Sensors
  - Correlation
  - Users
  - Domains

The main purpose of the EAM Web-based interface is to configure the EAM. It allows administrators to configure CMS information and the sensors that provide the events the EAM needs to forward to it.

Additionally, the EAM Web-based interface shows the status of the EAM, its conduits, and the sensors that provide them with events.

Reporting

The reporting in Tivoli Security Operations Manager is based on data that has been stored in the central database. You can either use the dashboard feature in the Web interface or the reporting engine to inspect your data.

The visuals in the dashboard are meant to provide a short-term, detailed view of events that are happening now and of historic events that are being investigated. Reports provide a longer term view that makes it easier to see trends.

Because reports are usually less detailed, they are easier for users to understand. This feature is useful to communicate with nontechnical managers and auditors who have responsibilities related to information security.

Tivoli Security Operations Manager uses JReport, a non-IBM packaged bundle, with the product to produce reports. Tivoli Security Operations Manager also ships with a number of predefined reports ready at your disposal.

Reports can be generated in real time or scheduled in advance. In either case, it is necessary to provide values for the report parameters, which are variables that are different for different reports, of the same type. For example, reports that

contain event information have a start time and an end time to determine which events are relevant.

You can schedule reports to run at the same time of the day, the same day of the week, and so on. This allows report generation, which can be a heavy task, to take place during times of low usage.

28.3 Physical components and architecture

In this section we focus on the general physical architecture of the components, from event gathering to analysis. We discuss three different deployment scenarios: single server, distributed, and high availability.

The deployment architecture for Tivoli Security Operations Manager consists of two main logical component areas:

- *Event Aggregation Module* (EAM) - The Event Aggregation Module performs the task of gathering data from the various network security devices, normalizing that data, and then filtering, batching, and transmitting that data to the Central Management System (CMS).
- *Central Management System* (CMS) - The Central Management System (CMS) acts as the hub for Tivoli Security Operations Manager, bringing together event data streams from all of the deployed EAMs. The CMS correlates this event data, performs deterministic threat analysis, calculates the threat posed to the destination by the event, and applies the rules configured in the stateful rules engine to the event stream, allowing the system to respond to specific attack signatures and events of interest. Both the real-time and persistent data present relevant information through the user interface and advanced analytic module.

28.3.1 Single server deployment

A single server deployment, as shown in Figure 28-9 on page 883, is by far the simplest configuration for an enterprise security architecture. A centralized, single-server Tivoli Security Operations Manager architecture provides several functional benefits:

- Centralized CMS and EAM modules on the same system.
- Ease of management with all components on same system.
- Ideal for small to mid-size, single-site deployments.

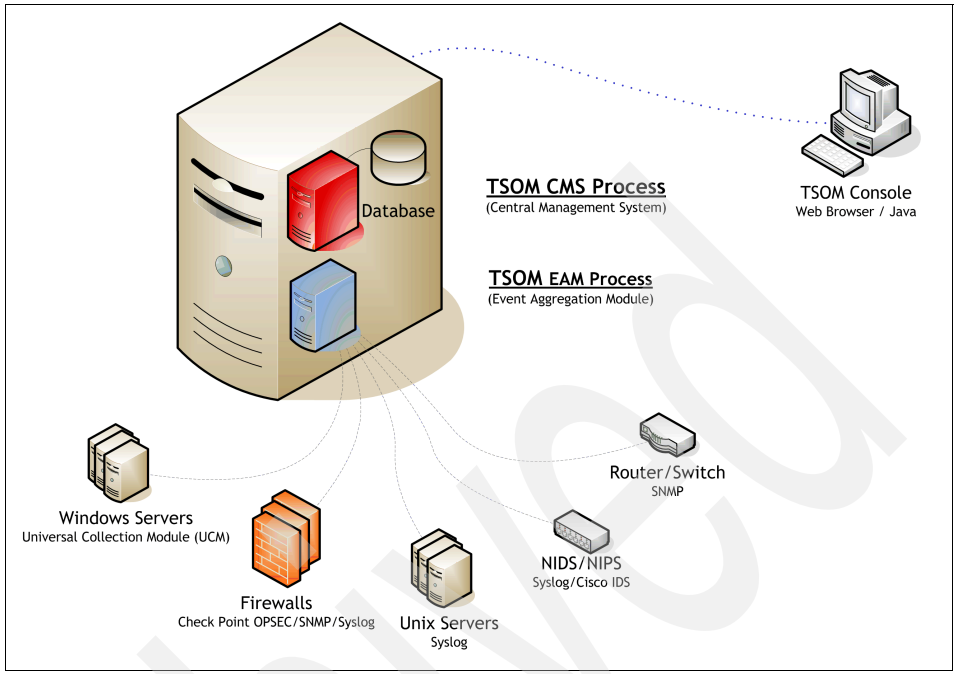


Figure 28-9 Single server deployment

A single server Tivoli Security Operations Manager deployment, is ideal for small to mid-size enterprise security architectures that may contain a few devices of each flavor (several NIDS, a few HIDS, and so on). This deployment is also ideal for architectures in which security information is maintained for one site, such as a branch department or the security department of a small to mid-size business.

Capacity wise, a single server deployment can handle a mixture of several NIDS, HIDS, and firewalls, with several server/system security event sources. With a properly sized and configured system, a Tivoli Security Operations Manager single server deployment should be able to handle ideally between 10 to 20 active security event sources (from a mixture of NIDS, HIDS, firewalls, and other event devices/sources).

28.3.2 Distributed deployment

A distributed deployment of Tivoli Security Operations Manager, as shown in Figure 28-10 on page 884, provides a standard configuration for mid-size to large enterprises needing to manage enterprise security architectures across multiple sites simultaneously. This deployment architecture allows for event aggregators (EAMs) to aggregate event information from multiple locations back into the CMS

for analysis. Some of the benefits of a distributed Tivoli Security Operations Manager deployment include the following:

- Multiple EAMs to handle large amounts of security events.
- Distributed architecture covering multiple sites.

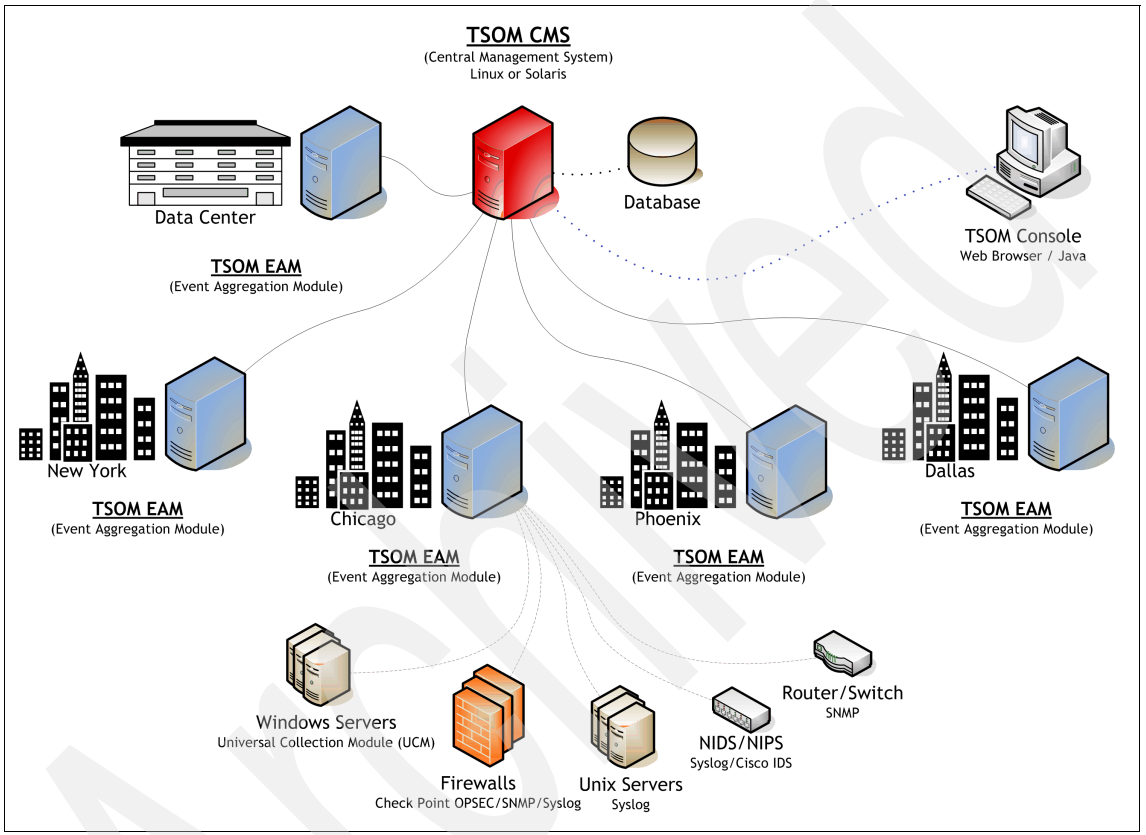


Figure 28-10 Distributed deployment

The distributed deployment consists of multiple EAMs located at multiple locations throughout the enterprise. Each of these local EAMs allow for the local consolidation and aggregation at each geographic site, enabling network traffic to be kept to a minimum, as well as providing a more secure mechanism for sending back events to a central location. With only the EAMs reporting back to the CMS at the primary location, security events are transmitted securely in an encrypted fashion from only several hosts, as opposed to potentially hundreds of devices reporting back to one central location. This tiered architecture also keeps failures in the system at a minimum.

28.3.3 High-availability deployment

A high-availability deployment, shown in Figure 28-11, provides high-availability features into the enterprise security architecture for mid-to-large sized businesses. By enabling high-availability (HA) features into a security architecture, enterprises can further minimize the risk of any potential failures to the enterprise security architecture. HA enables the Tivoli Security Operations Manager application to remain online, aggregating and correlating security event information even during a failure.

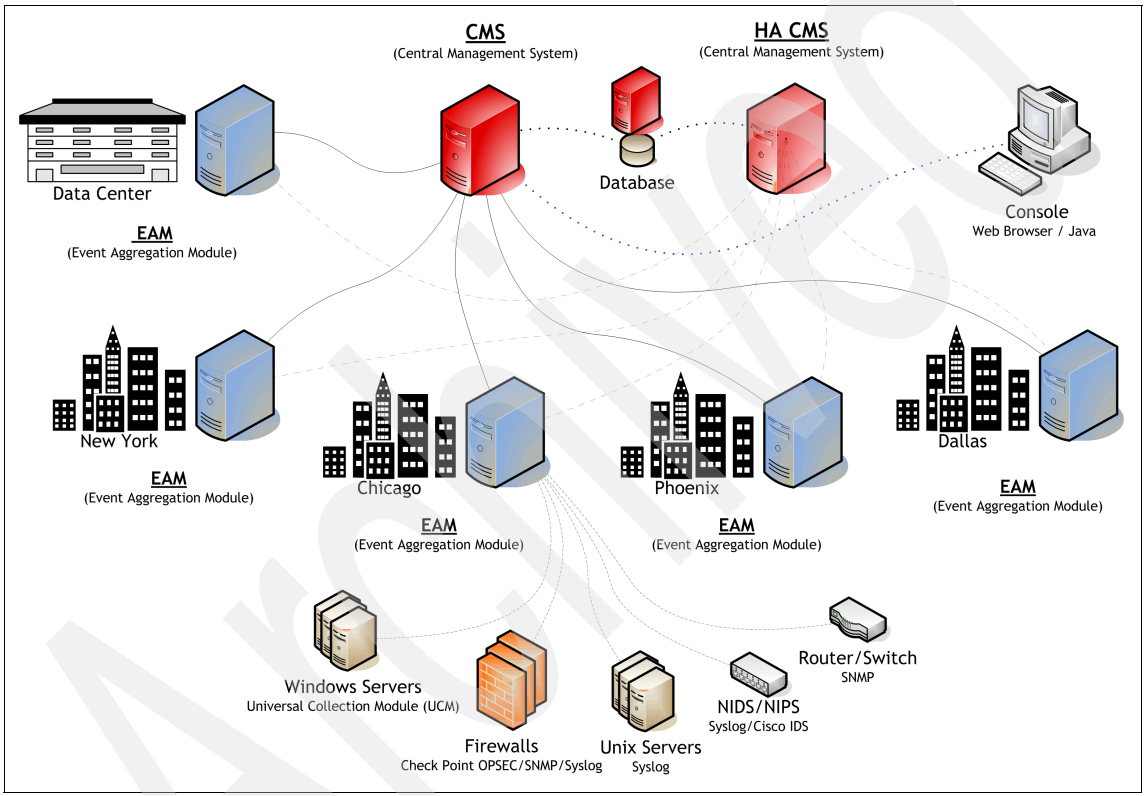


Figure 28-11 High-availability deployment

High-availability deployments provide assurance that the enterprise SIEM architecture remains active, even in the event of a failure. If the primary CMS were to fail, the EAMs automatically fail-over to the backup CMS. The high-availability mode is performed in an active-standby mode with one CMS as the active primary and the secondary CMS in standby mode.

28.3.4 Network placement

Let us also discuss in which network zone you should place the different Tivoli Security Operations Manager components. Figure 28-12 depicts an overview of a typical deployment.

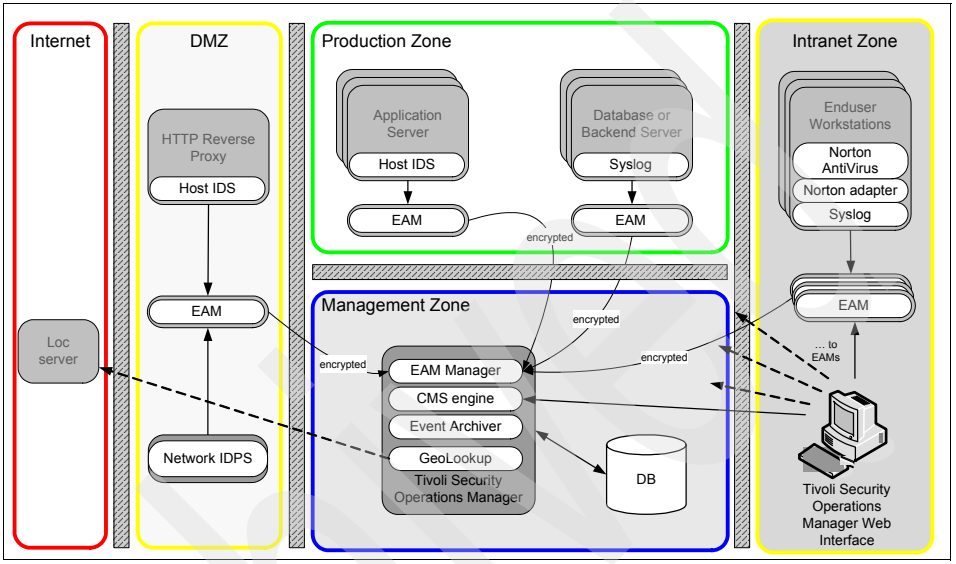


Figure 28-12 Network placement

The CMS (together with the Tivoli Security Operations Manager database) and one EAM should be placed into the management zone because they represent very crucial and detailed forensic data about your network and application traffic.

Other EAMs can be located in other network segments, typically as close as possible to the sensors to improve performance. You can also deploy multiple EAMs if the amount of collected events call for more than one EAM. Secure communication between these EAMs and the central EAM/CMS ensures data confidentiality.

The administrative Web interface can be located anyplace on the network. In Figure 28-12 we can administrator placement in the regular intranet zone. From there all tasks can be managed via secure communication to the CMS and all EAMs.

28.4 Conclusion

We introduced the general idea behind an enterprise *Security Information Event Management* (SIEM) system and why it is necessary today. We also introduced the logical components and architecture of Tivoli Security Operations Manager. We explained the processes (aggregation, normalization, contextualization, and correlation) that are used to centrally manage security events from disparate sources in a common way.

Next we described the major components that Tivoli Security Operations Manager uses (Event Aggregation Module, Central Management System, and Event Archiver) to process these events.

Finally we looked into the different physical deployment models on how you can integrate Tivoli Security Operations Manager into your environment.

29

Building a security information event management system

The purpose of the security information event management system is to address the data collection, analysis, and archival requirements of a computing solution to manage and measure the effectiveness of the security implementation. Security event analysis and reporting includes real-time review and management of events as well as after-the-fact analysis to anticipate and take actions to maintain and improve the integrity and reliability of resources. Security Operations Manager addresses both of these requirements. The Central Management System and reporting engine alert security managers to problems by correlating thousands of events into more specific incidents to identify attacks. The reporting engine provides near-term reporting and analysis of events in detail.

We look at the IT setup in order to describe the SIEM solution approach. The scenario depicts our already introduced small medium business (SMB) enterprise, Stocks-4u.com. In the next section we describe the scenario profile and an excerpt of their current IT deployment, which includes a basic Web infrastructure and a few security products that are implemented.

29.1 Scenario profile

Stocks-4U.com is an upcoming online trading company in the early stages of implementing an e-business infrastructure facing the Internet. The importance of having a properly deployed enterprise security architecture is critical. The CISO has come under serious pressure to ensure that no potential embarrassing data leakage events occur. As a result, Stocks-4U.com invested heavily in their information security architecture throughout their e-business infrastructure. The current architecture consists of a public facing Web site with three zones:

- Internet - Clients access the public facing Stocks-4U.com Web site from the Internet as they would any other site. Stocks-4U.com contains a direct customer portal, as well as a separate portal for partners. Clients and Partners from a wide variety of geographic locations access the site on a regular basis.
- DMZ - The DMZ, or Demilitarized Zone, contains the network perimeter that serves as the functions for border security through intrusion detection (IDPS), load balancing through a Layer 3 load balancer, and application security and caching through the use of an HTTP reverse proxy.
- Internal Production Zone - The internal production zone contains the production Web, application, and database servers that sit behind the DMZ. These servers power the Stocks-4U.com client and partner portals.

Stocks-4U.com's overall environment is heterogeneous with multiple hardware and software platforms. There are two firewalls deployed: Cisco PIX and Check Point FireWall-1. One Internet demilitarized zone (DMZ), one production zone, and one public facing internet zone in one physical location are set up.

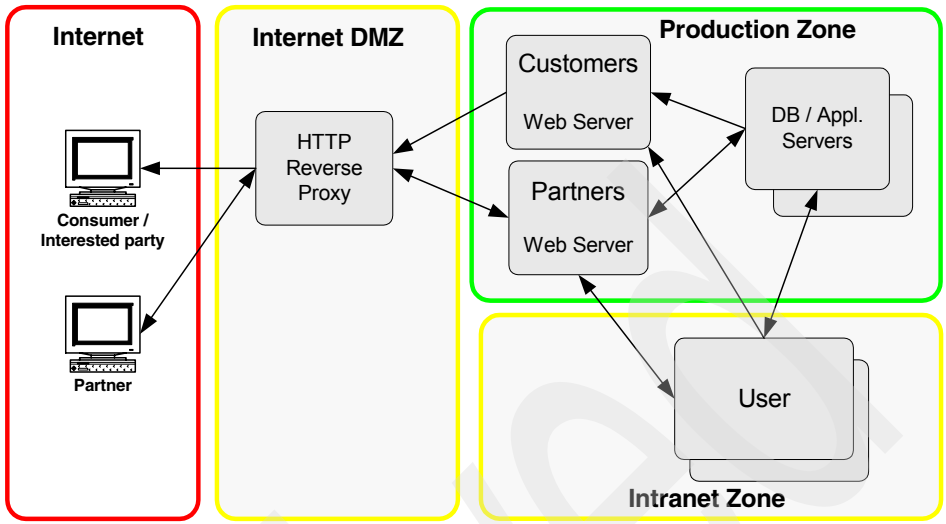


Figure 29-1 Initial IT architecture for Stocks-4U.com

29.1.1 Security-related problem

This company's infrastructure is very typical, on a smaller scale, of a popular Web site portal architecture. They have a small staff and limited technical resources. In their current configuration, the company is limited as far as the time and abilities of their technical staff to be able to respond and thoroughly investigate all security-related incidents as they occur. The company must rely in turn on vendor-based security solutions that are highly centralized and focused, but enable the company to free resources for other technical issues. Due to the nature of the information contained within each portal environment, the company placed security near the top of their priorities to minimize the risk potential as much as possible.

29.1.2 Business requirements

At the current stage, the CISO is looking for a way to validate and ensure the security of the network, and thus enhance customer and partner confidence. Having invested in a variety of enterprise security products (firewalls, intrusion detection systems, and routers), there is still a lack of comfort that the network environment is secure. After a competitor's recent information breach, upper management demanded that such an embarrassing situation never occur regarding the Stocks-4U.com Web site. The CISO in turn relies upon his IT staff for recommendations as to how they can change or improve the security architecture to ensure that such risks are minimized. The IT staff is requesting more tools and the manpower and skills required to manage these products. The

events recognized by the current security tools, such as invalid logons, attacks, and viruses, must be investigated and handled. With a small staff, it is difficult to handle all events, much less let the staff attend training or take vacation without constant pages and calls. While the CISO is concerned about the amount of investment, the need to prevent an embarrassing situation is further outweighed by cost. However, given all of the security products currently deployed he has no readily apparent way to measure the effectiveness of the solution.

29.1.3 Business design

To meet the needs of the business, the audit flow structure of the security event management system, depicted in Figure 29-2, must look at the audit events from the security tools be able to identify real threats and attacks and provide information as to actions that should be taken to the security staff. Identifying real threats from the volumes of alerts generated by multiple security sensors will make the environment more secure and more manageable. Predefined actions for specific event types will allow for quick and consistent handling of situations and increase the quality of service to users. The system must also be flexible enough to support additional tools and systems as they are implemented in the near future.

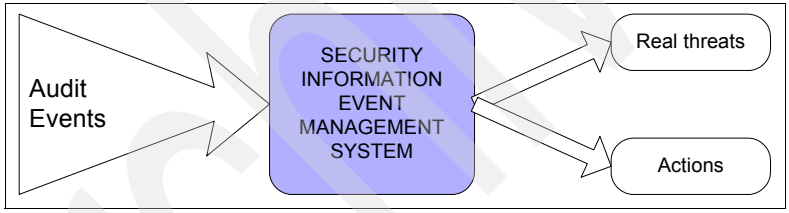


Figure 29-2 Audit flow structure

29.1.4 Security design objectives

The primary objective of the *Security Information Event Management* system (*SIEM*) is to enhance the security management function through the collection, analysis, and archiving of security data generated by the security environment on both real-time and historical modes. The SIEM has to be able to isolate real security alarms from the vast flow of security events and correlate events from several sources. The SIEM must also support the actual network structure, including firewalls, routers, and servers. A single control point to monitor, defend, and respond to attacks and intrusions is needed.

It is important to understand that the implementation of security tools does not eliminate the need for skilled security specialists and administrators. All tools must be configured to the specific system environment. The thresholds must be

tweaked. Automated actions may vary by alert source or target. The desire is to make the security specialist or technician more effective by delegating common tasks to operator-level personnel or even automating responses to common situations.

29.2 Security Information Event Management System

A security information event management system, or SIEM, is responsible for capturing, analyzing, reporting, archiving, and retrieving records of events and conditions.

Figure 29-3 shows a use case model of a SIEM architecture. The physical view shows the systems involved in the transaction. The component view depicts the information flow control function that will examine messages being sent and, based on a set of rules, will enable valid messages to flow. Invalid messages are stopped, and a record of the event is sent to the SIEM system. The logical view breaks down the aggregation and correlation processes into distinct functions.

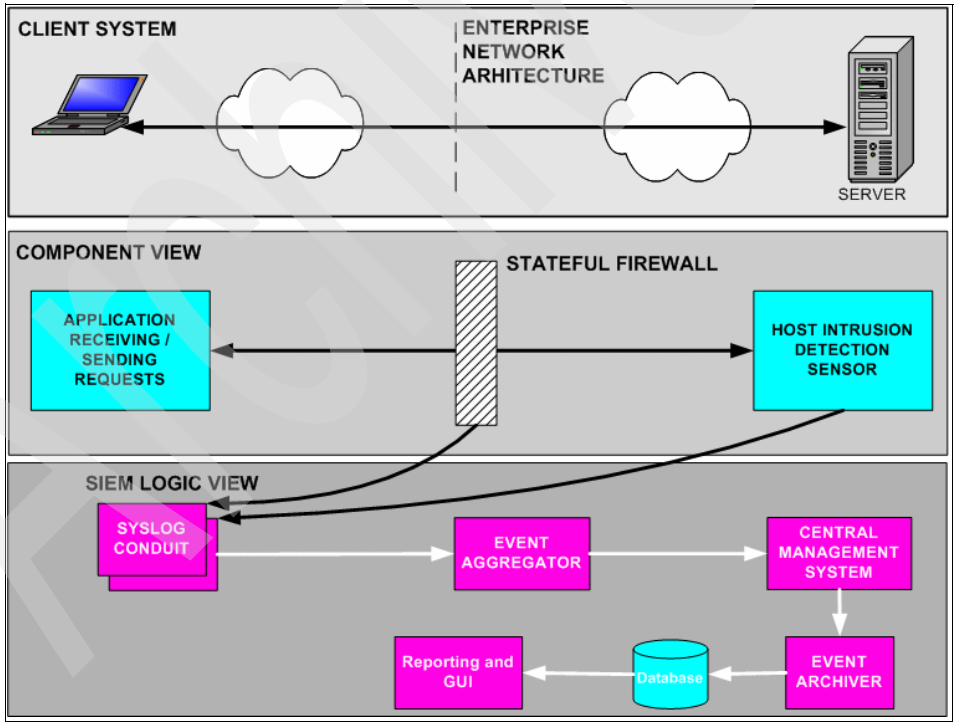


Figure 29-3 Physical, component, and logical views of a SIEM system

The *syslog conduit*, or log aggregator, is a standard aggregation point that collects logs from sources such as the system log file in a UNIX environment or the event log file on a Windows system. Most security products have a log handler function that generates events, such as a firewall violation attempt. Mostly, it routes them to a management console or stores them to a log file. Conduits are then used to retrieve this stored event information into the SIEM system.

The *event aggregator* converts or reformats the events from the conduits or log handler into a normalized format usable by the SIEM system. The event aggregator utilizes a rules-based knowledge base of the different message formats used in security device logs and messages. By mapping a particular message to an appropriate event class, the event aggregator can put disparate security events into a standardized event class and message format, normalizing the security events.

The *central management system* (analysis or correlation engine) receives events from the event aggregator. It utilizes a rules-based correlation engine and filters to correlate and analyze the events. This is the core function of the central management system, and its effectiveness depends on the *rules* defined and configured.

The output from the correlation engine is sent to the *event archiver* to store all records in a centralized *database* that will be used for reports generation and event storage.

The main graphical user interface and reporting engine executes extensive analysis on long-term data stored in the event database. These reports help understand general vulnerabilities within the environment that do not generate incidents.

Note: The *correlation engine* is an important part of the model because it receives events from several sources and correlates them. Security rules must be defined very carefully for rules-based correlation engines to work properly.

29.2.1 SIEM system at Stocks-4U.com

To apply the concepts of a SIEM system within their IT infrastructure, Stocks-4U.com would first need to identify which components in their configuration are generating security-relevant events. Next, a log handler, conduit, has to be configured centrally to collect events and forward them to an event aggregator. For those devices or systems unable to centrally forward their logs, a collection module is added to each of those components. In Figure 29-4 on page 895, we see the network diagram for Stocks-4U.com.

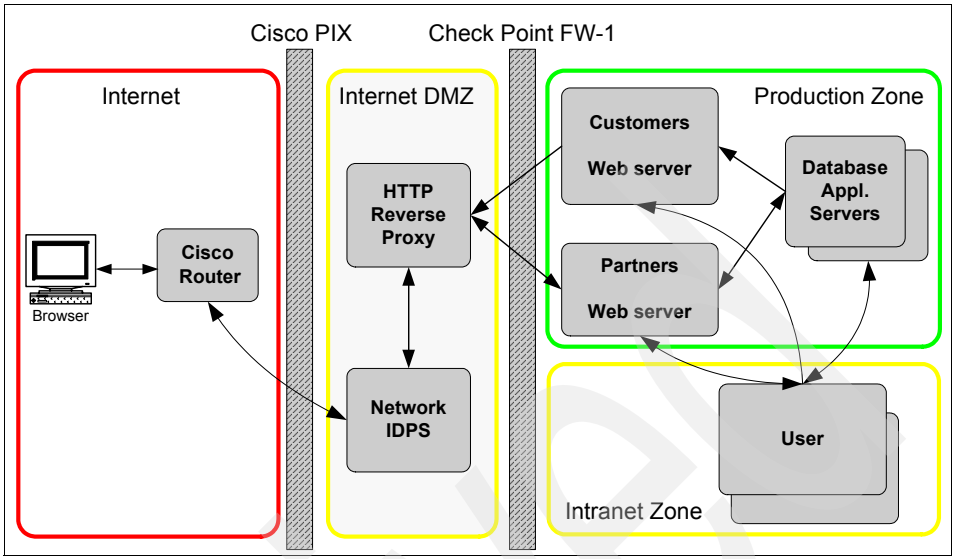


Figure 29-4 General network diagram

As shown, the components to have security event information collected are as follows:

- Cisco router: Collects configuration change data, connection information, and exception or error events.
- Firewalls: Collects information about flows as well as accepted or denied connections among parts of the network.
- Web servers: Collect information summarizing the activity of the Web server and events such as unsuccessful logons, configuration changes, or long URL attacks.
- Servers: Collect access control exceptions from the operating systems and from applications.
- Users: Collect data on viruses detected by antivirus software.

29.2.2 Integration of Security Operations Manager

As discussed in the previous chapters, the identity and access control components of our security architecture show how the consolidation and automation of functions provide effectiveness and efficiency in an overall solution. The same applies for the security information event management system. Multiple security management consoles that are spread across multiple zones in the environment do not support a truly secure setup. To implement security information event management at Stocks-4u.com, the central

management functions should be grouped into a single secure zone. In Figure 29-5, the Security Operations Manager CMS and a central EAM are placed within the management zone. Multiple *Event Aggregation Modules* are placed in the Internet DMZ, the production zone, and the intranet zone, which use encrypted communication through the firewalls to send collected information to the *Central Management System*. We are using distributed EAMs to balance the load on collecting event information before sending the information to a central EAM Manager.

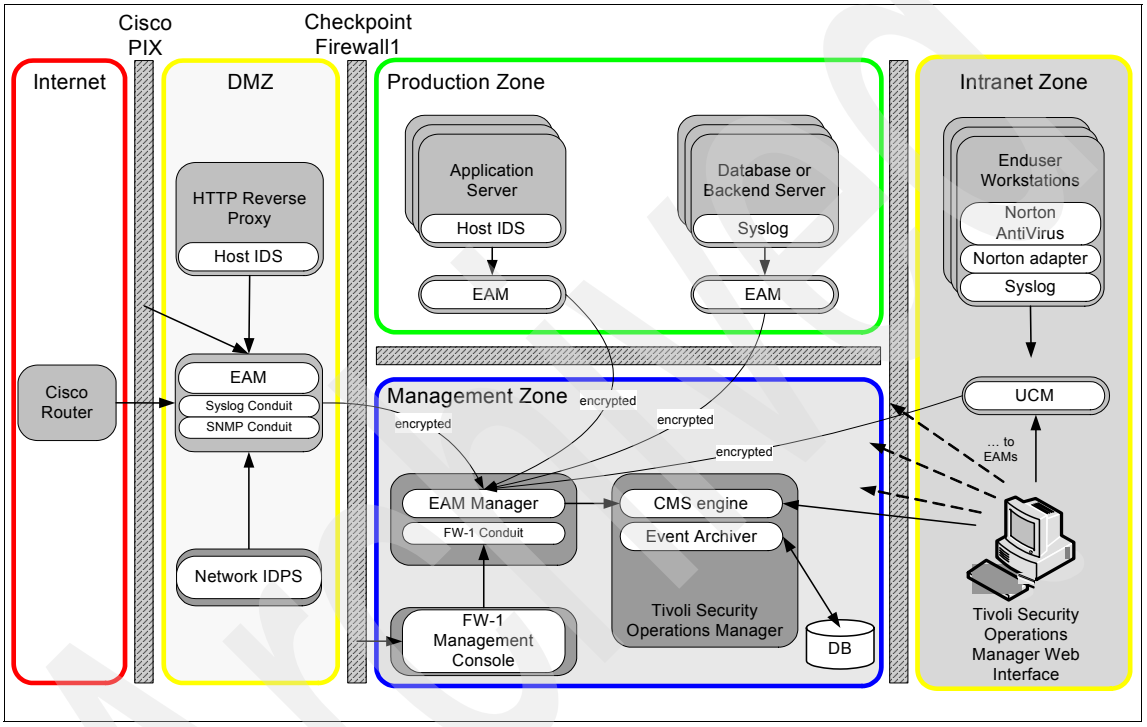


Figure 29-5 Detailed Stocks-4U.com security architecture

Environment

The Security Operations Manager server requires that the application be properly installed and configured. In addition, for enterprise deployments, it is highly recommended to use a separate server for housing the central database.

Communications with the Security Operations Manager server

In this section we discuss the different ways Security Operations Manager and the other products exchange security event information.

Cisco router

The Cisco router generates either Simple Network Management Protocol (SNMP) traps or SYSLOG messages. In this case, when the configuration changes, system-level errors occur, or when there are unsuccessful logons. The SNMP event or SYSLOG message is sent directly to an event aggregator known as the *EAM*. Running on the EAM is a set of conduits, one for SYSLOG as well as for SNMP. The SYSLOG conduit consists of an application that captures and forwards the SYSLOG messages, from the Cisco router, in this case, to the central EAM Manager, which then forwards the messages to the Central Management System server for correlation.

Cisco PIX

The Cisco PIX Firewall sends SYSLOG messages, in this case, or SNMP traps to the EAM, which has a configured conduit. A conduit is configured to be made aware of the devices configuration (IP address, host name, and so on), in order to tie specific messages to a specific event or device source.

Check Point FireWall-1

The Check Point FireWall-1 sends its logs to a dedicated management console (data transfer is encrypted). The EAM uses a native Check Point OPSEC interface (the FW-1 Conduit) to receive the log messages from the FireWall-1 management console.

Host Intrusion Detection System (Host IDS)

Security Operations Manager can map Host IDS-based events, which are detected and logged by the Windows or UNIX system logs, into relevant security-related incidents. The EAM receives SYSLOG messages from UNIX systems via the SYSLOG conduit or Windows Event Log messages from Windows systems via the *Universal Collection Module* (*UCM*) and the *XML Conduit*.

Norton AntiVirus

Norton AntiVirus writes events in the Windows system event log. The UCM recognizes the virus-related events sent by Norton AntiVirus on a Windows system and forwards them to the EAM, which normalizes the events as Windows-system related security events and those from Norton AntiVirus. These events are then sent to the CMS for further correlation.

Web Management Console

The Security Operations Manager administrator(s) can work on their regular workstation, located within the intranet. The Web Management Console connects to the distributed EAMs and the CMS using secure communication.

29.3 Expanding security monitoring

To enhance the security of Web environments, other security tools should be installed. For access control functions, a Web security server solution such as Tivoli Access Manager WebSEAL or a Tivoli Access Manager Web server plugin is recommended. In order to monitor network traffic (users, customers, and partners), a network intrusion detection and prevention system such as IBM ISS Proventia is needed. Because most of the suspicious activity and threats still come from within the enterprise, a probe within the internal part of the network is essential as well. Both of these components are integrated easily into the Security Operations Manager SIEM system.

HTTP reverse proxy server

An HTTP reverse proxy server provides single-point management of authentication and access control. Security Operations Manager is capable of receiving security event information from Tivoli Access Manager's reverse proxy server, WebSEAL, in order to provide higher-level correlation. Security Operations Manager is also capable of receiving security event information from Tivoli Access Manager's Web server plugin component.

Intrusion Detection and Prevention System

An Intrusion Detection and Prevention System (IDPS) is a detailed packet analysis system designed for computers (Host IDS), Web servers (Web IDS), and networks (Network IDS). An IDPS gathers and analyzes information from various areas, either within a computer or a network, to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). An IDPS uses a vulnerability assessment (sometimes referred to as scanning), which is a technology developed to assess the security of a computer system or network. Intrusion detection functions include:

- Monitoring and analyzing both user and system activities
- Blocking attacks as they come down the wire
- Analyzing system configurations and vulnerabilities
- Assessing system and file integrity
- Recognizing patterns typical for attacks
- Analysis of abnormal activity patterns

29.3.1 Security Operations Manager resources

For information about the IBM Tivoli Security Operations Manager product, visit the following Web site:

http://www.ibm.com/software/tivoli/products/security-operations-mgr/

For information about other Tivoli management products, visit the following Web site:

http://www.ibm.com/software/tivoli/sw-bycategory/

Security Operations Manager supports the following types of applications:

- Firewalls
- Web servers
- Intrusion detection and prevention systems
- Antivirus products
- Routers
- Operating systems log file events

29.4 Mapping the solution to the organization

The ability to delegate the monitoring and audit functions within Security Operations Manager enables Stocks-4u.com to distribute responsibilities to different administrative people. Security administrators have the responsibility for customizing the rules files and defining details such as thresholds and categories, while IT operators only see basic security alerts and events. This functional delegation model is applied according to the individual internal organization of the company.

The purpose of this discussion is to describe the functional responsibilities.

Figure 29-6 on page 900 depicts the Stocks-4u.com organization and the role of each factor:

- Security administrator

  A *security administrator* defines the audit policies, such as which system should be audited, which are the trusted hosts, and so on. This job also configures Security Operations Manager and defines values such as thresholds, categories, adapters, and so on, to fit into the company profile and needs. Another administrator task is to document security instructions that describe situations and what actions should be taken for specific events, and to build automated scripts when possible. This is an ongoing task, as new threats are always being discovered and new tasks are needed to protect the network.

- Operator

  An *operator* sits in front of the central console and receives the security audit events. The job is to react accordingly and apply the procedures and documents written by the security administrator. An operator interacts with

the system administrator, the application administrator, or both to solve the problem, and could also interact with the users.

- Support

  This function can be an external product's *support* or even the security administrator. The major task of support is to assist the operator in the problem resolution by performing tasks an operator is not authorized to do.

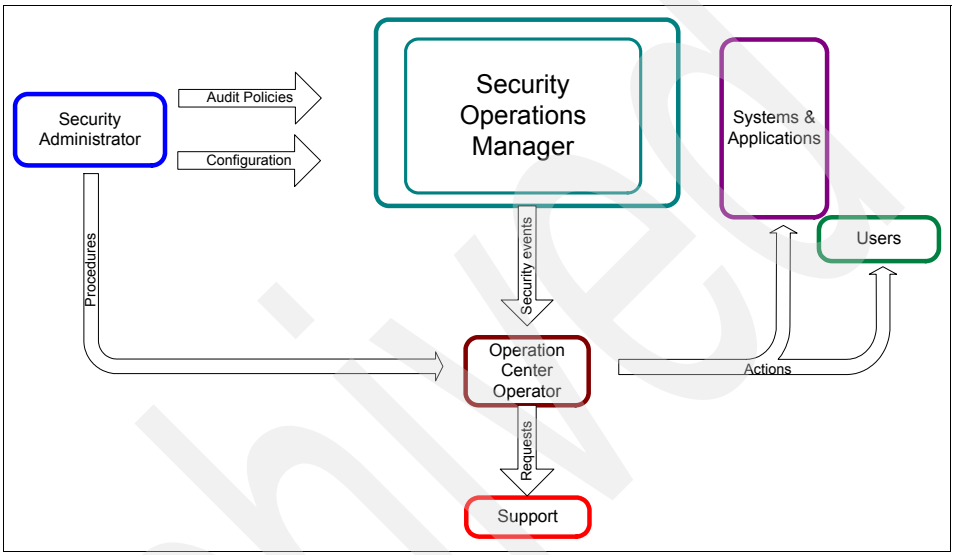


Figure 29-6 Organization flows

This ensures the continuity of security management without requiring the highest skilled administrators to perform the day-to-day management tasks. This helps the Security Administrator avoid common tasks in order to focus on upgrading skills to increase the security level and awareness of the company overall.

29.5 Summary

It is the function of the security audit subsystem to collect alerts from a variety of sensors, analyze them, identify real threats, and, if necessary, perform some automated actions. These actions can include displaying an alarm, executing a script, shutting down part of the network, and closing a port or blocking an IP address on a firewall.

Security Operations Manager provides a simple, easy-to-use enterprise console to monitor, view, and manage alerts across the enterprise. By correlating events from multiple security tools, Security Operations Manager can recognize attack

patterns and escalate events and incidents to the console. Because the events are routed to a single point of control (the management area), fewer resources are required. In addition, because the rules and actions have been defined by the administrator, lower-level personnel monitor the console and handle basic alerts.

In addition to the real-time handling of events, Security Operations Manager also has a powerful reporting tool for near-term reporting and for trend analysis to facilitate preventive measures and planning.

30

Compliance management with Tivoli Security Compliance Manager

IBM Tivoli Security Compliance Manager is the IBM security policy compliance management product that acts as an early warning system by identifying and reporting security vulnerabilities and security policy violations for small, medium, and large businesses. IBM Tivoli Security Compliance Manager makes sure, by deploying predefined policies based on best practices, that all servers in an enterprise meet the policies and regulations that the enterprise is subject to comply with. It also ensures that the gathered information gets to the right people so proper actions can be taken.

30.1 Business context

After providing a short definition of *security compliance management*, we describe the factors that influence why and how compliance management should be conducted in a given business context. Further, we explain the general business requirements for a security compliance management solution and give some recommendations on how to put into practice compliance regulations.

30.1.1 Introduction to compliance management

The process that ensures that the security, regulatory, and operational policies of a company are adhered to is called *compliance management*. It requires the ability to report on the current compliance status of security controls of any installed system and to react to any observed deviations.

Security controls exist on the technical, process, and organizational level:

- An *organizational level* security control can be defined as a type of control that affects or depends on the role of an employee as part of an organization. An example can be a concept like *separation of duties*, that is ensuring that someone changing something is not the same person controlling the business need and proper execution of the change. This type of security control may require an organizational setup where those two employees report to different managers.
- A *process level security control* is defined as a series of steps that have to be executed to make sure a certain condition can be met. An example could be a concept like the *four eyes principle*, where a specific authorization requires two signatures (or passwords) to be presented before a transaction can be completed. As a result, this process step would always require two employees to be available for execution.
- A simple *technical security control* is concerned with a specific implementation of a service in the security context. For example, a minimum required length for a password or specific permissions that are defined for accessing an operating system resource or business data. Operating systems and applications provide configuration settings that allow the administrator to specify minimum password lengths so that the system itself will enforce this control.

  A more complex *technical security control* can be the requirement to make sure a system meets all the necessary prerequisites to be a part of a secure network, for example, run an anti-virus service (with up-to-date virus definition files, of course) and have a firewall turned on.

While it can be hard to have process level or organizational level security controls checked automatically (by a computer), technical security controls can be

automatically monitored, as this only requires collecting configuration parameters (for example, minimum password length) and comparing these with predefined desired values.

IT security compliance management is about ensuring that the defined settings (in a security policy or standard) are implemented correctly and consistently on all the installed IT systems.

Because in practice there can be reasons why a specific configuration setting cannot be enforced in the desired way on a number of systems of each type (usually due to an application either explicitly requiring the parameter to be set differently or because it is simply not working otherwise) a significant part of compliance management is handling exceptions to the defined security policy or standard.

30.1.2 Why compliance management

Information currently is the most valuable asset of a company and therefore must be protected accordingly. The aftermath of previous security-related incidents where private information was disclosed, was catastrophic to the companies, losing not only their credibility but also sustaining financial losses.

Compliance management is key not only to the security officer in an organization, but to the rest of the management team:

- The Chief Information Officer has to make sure all systems are protected and comply with regulations for technical, security, and legal purposes.
- The Chief Operations Officer has to make sure that all systems are up and running so the operation of the company does not get disturbed.
- The Chief Financial Officer must guarantee the safeguarding and use of a company's finance; therefore, a loss or fine due to compliance issues is important.
- The Chief Executive Officer can be legally liable if compliance regulations are not implemented successfully and a breach occurs.

In short, compliance is a main issue throughout the company, from the technical level all the way up to the most executive positions.

Most businesses today heavily rely on their IT systems, and damage incurred to their critical systems through downtime can take a company out of business for a period of time. Therefore, minimizing the risk becomes a best practice, both for the technology and for the business aspect.

Through regulation (for example, Basel II[1] in the banking sector), the excellence of risk management for IT systems, which is part of the operational risk complex,

even has an impact on the competitive advantage of banks because it can affect the interest rates a bank can offer its customers.

Because the configuration of security relevant settings in an operating system has a direct impact on the resilience of the system against attacks, viruses, worms, or computer criminals, ensuring these settings are always at the desired level directly lowers the risk to the system.

Due to current legal regulations in some US states[2], companies must disclose if they had a security incident in which information that contained private data was disclosed. As time passes, it will become more common for states and countries to pass laws to enforce this. Due to these new regulations, companies can be greatly exposed. To reduce this exposure, involved systems must adhere to compliance in order to lower the risk.

Further, checking the security controls of managed systems ensures that a system does not *degrade* in its security controls posture due to changes on the system after it is installed. For example, changes made while resolving a problem, while installing or upgrading a new application or middleware, or due to an attacker changing the configuration to hide his tracks or to compromise the system.

[1] *Basel II: International Convergence of Capital Measurement and Capital Standards: a Revised Framework, June 2004* (more information can be found at http://www.bis.org/publ/bcbs107.htm)

[2] For more information about states that passed these kinds of law visit the following Web location: http://www.consumersunion.org/campaigns/Breach_laws_May05.pdf

Being compliant versus being in control

If you have ever been audited (or audited someone), you probably know that there is a difference between being:

- *In compliance*: All your systems and processes are operated and delivered according to the security policies and standards (and you have evidence for that).
- *In control*: You know what is in compliance and what is not, you know why, and you have a plan of action.

Now, what is more important? Being *in control* is. Because you could be in compliance by accident. Further, if you are compliant, but not in control, chances are high that you will not stay compliant for very long.

If you are in control, you will end up being compliant eventually. Or at least you will have it on record why you are not compliant.

And if you are not compliant and not in control, gaining control should be your primary goal.

30.1.3 Determining the how: influencing factors

While having security compliance management in place is generally a good security practice, there are several factors that influence if and how compliance management is implemented in a specific environment. Let us take a look at the main *dimensions* of compliance management.

- Frequency of checks

  How often is a compliance check being done? This does not only define how often the configuration data is collected from the systems, but also the frequency in which system administrators are called upon to fix or investigate identified deviations.

- Number and selection of controls

  Which and how many controls are checked? Are only operating system level controls checked or are application level controls checked as well? Which operating systems, middleware, and business applications need to be supported?

- Follow up time frame

  How fast do you have to fix reported deviations in the security configuration? This is a critical part of any compliance management solution that has to deal more with the follow up process of fixing a compliance issue than with the compliance check itself.

- Organizational and process checkpoints

  There is a particular need for *separation of duties,* for example, when the employee checking the configuration must not be the administrator of the system, and for process requirements, especially in the area of exception management and escalation if deviations are encountered or not corrected in time.

The factors that define *how much* compliance management, as defined by the dimensions above, has to be done are influenced by the threats in the external environment of a company. Let us summarize the external environment factors.

- Economy

  In which industry is the business operating? Is corporate espionage an issue? Does the company use outsourcing services? How dependent is the business on its IT systems?

- Regulatory/legal compliance

  In which countries and in which industry is the business operating? Which regulatory requirements exist that have an influence on required operational risk and the level of IT security? What level of scrutiny is executed by the regulators? It is useful to keep in mind that an IT security compliance management system can provide a lot of evidence for executed control.

- Attacks on IT systems (targeted or random)

  The main reason why IT security compliance management is a good security practice today and should even be considered a mandatory task when using IT systems at all is that businesses usually cannot afford successful attacks against their IT infrastructure. The threats against IT systems have become so *advanced* that one does not even have to have *enemies* to become subject to an attack, because many attacks are done automatically by worms and viruses. Even if critical systems are not directly compromised, a single infected system in a company network will negatively affect other systems and incur costs for the clean up.

Next, let us look at the internal environment factors of a company.

- Business and IT processes

  The value and amount of (business) information processed defines the level of security the processing system requires. And because security is always about the weakest link, related infrastructure systems need to be protected too.

- Organization

  The size and setup of the organization, for example, defines the speed of the reaction to deviations from the desired security level. Further, it will have a

significant impact on the requirements of an IT security compliance management solution, such as the administration approach.

- Technology/existing IT environment

  The existing IT environment defines the scope of the operating system, middleware, and business applications that need to be supported by any IT security compliance management solution.

In mature businesses, these influencing factors have shaped the existing security policies and standards as well as work practices or procedures:

- Security Policy

  Non platform specific or high level security requirements.

- Security Standards

  Platform specific controls (for example, configuration settings).

- Practices/Procedures

  Platform specific or non-specific descriptions on how to implement the security controls, for example, process steps, required documentation templates, and so on.

Further, these may have resulted in the IT department defining or creating the following tools to consistently implement the given standards and practices:

- Standard image/build

  Pre-configured installation image of an operating system with the correct settings applied. Note that these setting only apply to a specific time and must be revised periodically for new security updates that affect the standard image. After using any image, the administrator must check for new updates that may need to be applied since the image was created.

- Checklists

  Configuration or system activation checklists for configuration settings or tasks that cannot be predefined using an image. Checklists usually exist for all sorts of IT assets, from physical servers and clients with their respective operating system builds, to applications and complex environment configurations. The use of a security compliance solution might help to automate the checking of this settings for specific servers or applications.

30.1.4 General challenges

Now, even if the goal for security compliance is clear, defined by precise policies and standards (which often do not exist or are worded in broad, technically vague terms), the task of compliance management of a larger number of

systems bears the following major challenges in addition to the requirements resulting from the factors discussed above.

- Maintenance of compliance over time

  Even in a *stable* environment, systems are constantly changed because patches must be applied, updates must be installed or additional packages require a change in configuration of the underlying operating environment. Therefore, one important issue is the constant update of the compliance checks that are done on a system to make sure that all recent updates and patches are also being checked for.

- Complex environments

  Few businesses can claim that their environment is homogenous and centralized. Heterogeneous, geographically distributed systems in large numbers is the norm, with not only systems from multiple vendors, but also running several different versions of operating systems at the same time.

30.1.5 Some business conclusions

As a result of the influencing factors discussed above, a security compliance management solution must provide a flexible framework that can be configured and customized to the specific business in question. However, requirements for compliance management often result in functional or non-functional requirements for the technical solution *and* for the processes and organization behind the solution.

Let us look at a few examples.

- A *high frequency of compliance checks reduces the window of opportunity* for a potential attack/incident because the time frame that a vulnerability exists because of a control deviation is reduced. If the solution and the process to notify the system administrator is not automated properly, a lot of effort may be wasted in checking the reports that are generated in fast order.
- A *centrally maintained system for gathering and processing the compliance data lowers the cost of maintenance* when compared to a distributed system. However, it should be ensured that (the distributed) system administrators have direct access to the data of *their* systems to easily control the status of their system, for example, after a change. The need to request the information from a central team would be a burden on the central team and discourage the system administrator from proactive checks.

  As a consequence, the compliance management solution must allow for fine grained access control definitions so that system administrators are limited to the data on their systems only.

- While the ability to collect data on as many controls on as many platforms as possible sounds like the number one priority for a compliance management system, it should not be underestimated how important the reporting capabilities can be, especially if reports on the compliance status are required for legal/regulatory and audit purposes.
- Perhaps most important, it is necessary to realize that *business as usual* for compliance management systems is the management of exceptions from the defined standards (for example, because of conflicts with applications). Therefore, effective and efficient exception management should be on the top of the list of requirements for a compliance management solution.

At the end of the day, security is about the weakest link and, because of this, it is more important to have a consistent (if small) set of security controls in place on all the operated systems in a company, controlled through a reliable process in a reasonable time frame, than monitoring a hundred controls on a few systems in headquarters whenever someone feels like it.

Next we will look at the logical components of the Tivoli Security Compliance Manager solution.

30.2 Logical component architecture

The logical components of IBM Tivoli Security Compliance Manager (ITSCM) may be grouped in five different areas of responsibility, with the Security Compliance Manager server being the central component, as depicted in Figure 30-1 on page 913. The areas are as follows:

- Data collection components that build a framework for collecting security relevant configuration data from connected systems, such as operating systems, middleware components, applications, and so on.
- Administration components consisting of a graphical user interface and a command line interface are used to manage the Security Compliance Manager components.
- Compliance reporting components deliver different kinds of configurable reports for audit purposes and correcting deviations. The reporting engine is based on IBM DB2 Alphablox.
- Compliance evaluation components consisting of Security Compliance Manager snapshots and policies verify security compliance centrally. Both components are stored and maintained in the central database in order to ease the process of policy maintenance.

- The Security Compliance Manager server is the central component of a Security Compliance Manager infrastructure. Among the responsibilities of the server are:
  - Manages when the security compliance data is collected and which clients collect what kind of data using the data collection components.
  - Determines what security compliance data is collected, and how to interpret the data using the compliance management components.
  - Stores the security compliance data received from the clients and provides the available data to users through the administration console and administration commands.
  - Provides security violation details as a basis for the compliance report components.

The following sections describe the components of the five layers in more detail.

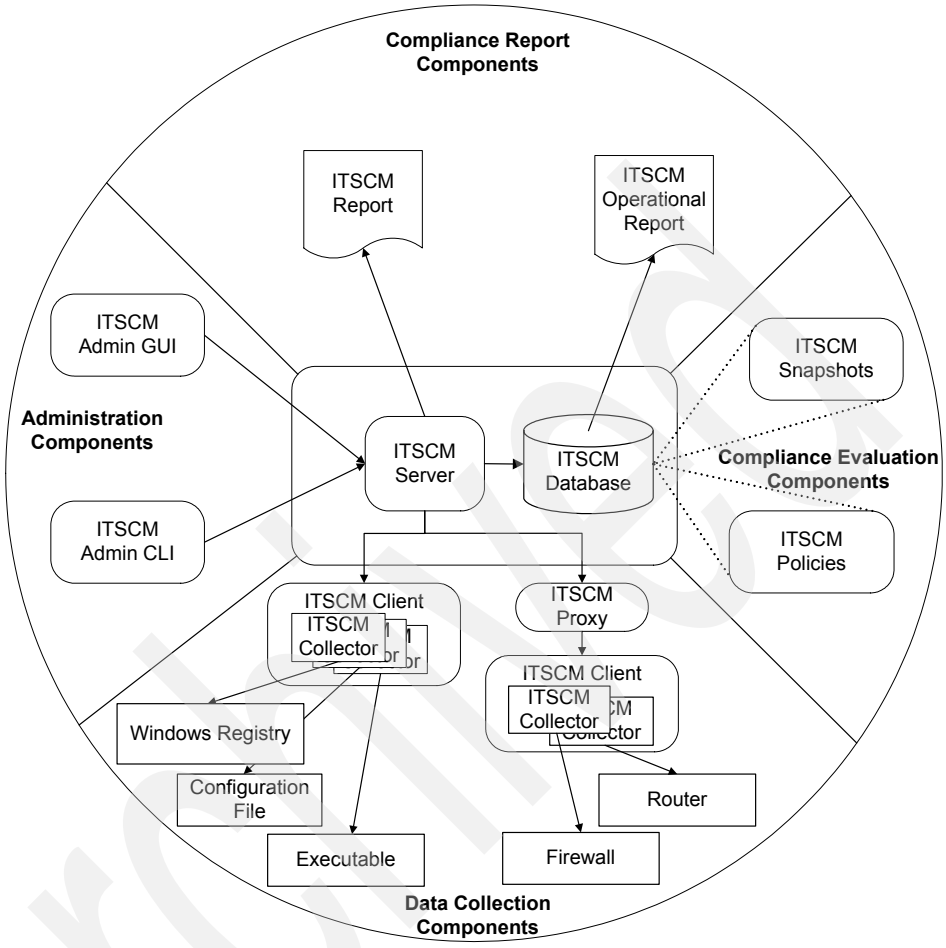


Figure 30-1 IBM Tivoli Security Compliance Manager logical component architecture

30.2.1 Data collection components

The data collection components are mainly responsible for collecting compliance data according to a schedule provided by the Security Compliance Manager server. One of the data collection components (the client) needs to be initially deployed to the systems that are to be monitored, either manually or by any other established means of software distribution in your environment. From that moment on, all components are centrally maintained using the Security Compliance Manager server management functions.

The data collection components are:

- Client
- Collector
- Proxy relay

Security Compliance Manager client

The *client* is Java language-based software that runs on systems to be monitored for security compliance. There are two types of clients: a push client and a pull client. A push client can establish a Secure Sockets Layer (SSL) connection to the server and send data. A push client permits communication with the server to be established by either the client or the server. A pull client must wait until the server establishes a persistent SSL connection with the client before data can be sent. Pull clients are considered more reliable and they are generally needed when the server is located behind a firewall when inbound connections to the server are not permitted.

Security Compliance Manager clients and client groups

A *client group* is a container used to group one or more clients together. Clients can be members of one or more client groups. The client group concept supports organizing large numbers of clients into categories representing operating system types, security policies, physical location, business objectives, or any other logical grouping.

Group inheritance

Adding policies and collectors to client groups is a powerful feature because of group inheritance. Every client that is a member of the client group, or a member of a subgroup of the client group, inherits the collectors and policies added to the group.

Client-server communication

Each client is uniquely identified to the server using a client identification number. Clients can be categorized into one of three types. Table 30-1 on page 915 describes the client types.

Table 30-1 Security Compliance Manager client types

Client type	Description
Push client	The push client permits communication with the server to be initiated by either the client or the server. Usually, the push client establishes an SSL connection to the server and sends data or asks for updates. The server only establishes a connection if an administrator forces an action to be performed on the client using the administration tools. Push is the default method to connect clients, as it requires less resources on the server.
Pull client	A pull client must wait until the server establishes a persistent SSL connection with the client before data can be sent. There are two situations requiring pull clients: ► The pull method allows clients to connect to a server, which is located behind a firewall that denies incoming connections. ► Clients located behind a Security Compliance Manager proxy relay need to be configured as pull clients. The pull mode operation uses more resources on the server and it is usually more reliable.
DHCP push client	A DHCP push client has a dynamic IP address that permits communication with the server to be initiated by either the client or the server. This option is used for systems that frequently change their host name or IP address. The general communication for the DHCP push client works just like the regular push client; the difference is the DHCP push client establishes the SSL connection.

After a connection between the client and the server has been made, either can send data to the other. Clients contact the server at periodic intervals called a *heartbeat*, which is every 10 minutes by default, to check for updates. This interval can be changed if necessary. During this heartbeat, the client receives any new or updated collectors from the server, along with any new or updated collector schedules and parameters. The client component software itself can be sent by the server and the client updates itself and restarts. This client/server heartbeat can be initiated from the administration console using the *soft reset* request function, bringing a client into sync without explicitly waiting for the heartbeat. Data gathered by the collectors that have run on the client is queued for delivery to the server on a more frequent basis, which is every minute, by default. Each client is uniquely identified to the server using a client identification (*CLI_ID*) number.

Securing the Security Compliance Manager client

The client is designed to provide a maximum level of security. It provides the following security features:

- Tamper resistance

  The Security Compliance Manager client is designed as a self-contained component. Each client contains its own Java Virtual Machine (JVM). For all operating system platforms other than HP-UX and NetWare, the JVM is automatically installed under the Security Compliance Manager client's base install directory. Access to the client files requires privileged access rights on the system in order to prevent misuse. This is extremely important if the client is installed on critical systems like firewalls.

- Secure communication

  The client establishes communication links with the Security Compliance Manager server based on the server's SSL certificate and IP address. Any other communication requests are denied. This assures that only the authorized Security Compliance Manager server is able to perform configuration requests like collector deployment or schedule changes. The server presents its SSL certificate during the first communication with the client (first contact trust). This certificate is used to verify the server's unique identity and to encrypt all traffic within the Tivoli Security Compliance Manager environment.

  When installing pull clients you can pre-deploy the server's SSL certificate information by copying the SCMHOME\client\ client.security file from a client already registered in the trusted Security Compliance Manager server. This avoids the opportunity for an unauthorized server to create the initial connection to a pull client.

Collector

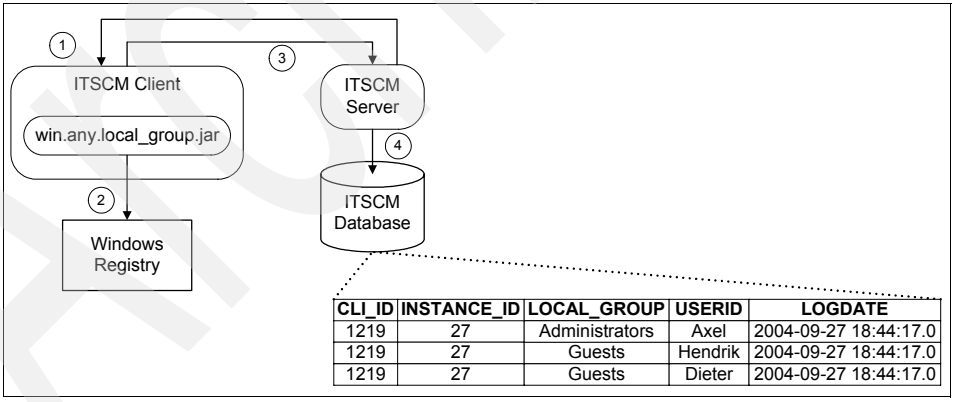
A collector is a Java language-based software module, packaged as a Java Archive (JAR) file, that collects specific information from a client system. A collector is designed to have a short execution time and to be non-invasive. The collector may use different methods for collecting data depending on the compliance data to be gathered:

- Reading the content of one or more files on the client system.
- Running an operating system command or utility and examining the output.
- Running an executable program packaged as part of the collector JAR file and examining the output.
- Reading information from the registry on Microsoft Windows systems.

- Remotely logging in to another system and gather data. This method allows you to collect security compliance data from systems that do not support Java applications.

Figure 30-2 depicts the concept of Security Compliance Manager collectors. The first time a collector is deployed to a client, the JAR file for the collector is sent from the server to the client, along with the collector schedule and any associated parameters (1). Multiple instances of a collector can be deployed to a client. Subsequent instances of the collector share the same JAR file, but run on their own schedule and with their own parameters. Each instance of a collector is uniquely identified by a collector instance number (INSTANCE_ID). According to its schedule, the collector starts to read security compliance data from its corresponding data source, for example, the Windows Registry (2). Data collected by each collector instance is queued by the client and delivered to the server on a periodic basis, by default every minute (3). Delivery of collected data is determined by two configurable settings in the client.pref file: flush.interval (the default is 60 seconds) and flush.threshold (the default is 100 messages).

The collected data is not stored on disk, but kept in memory until the connection to the server is established. The server stores the information received from the client into one or more tables in the database. The data in the database table is uniquely identified by the client identification number (CLI_ID) and the collector instance number (INSTANCE_ID) (4). When a collector instance is removed from a client, any data associated with that instance of the collector is removed from the database tables by the server.



CLI_ID	INSTANCE_ID	LOCAL_GROUP	USERID	LOGDATE
1219	27	Administrators	Axel	2004-09-27 18:44:17.0
1219	27	Guests	Hendrik	2004-09-27 18:44:17.0
1219	27	Guests	Dieter	2004-09-27 18:44:17.0

Figure 30-2 Security compliance data stored in collector-specific database tables

Securing the collector system

The Security Compliance Manager collector system provides security features to prevent unauthorized manipulation of deployed collectors and the deployment of collectors that are not appropriate for a particular environment. Figure 30-3 shows the signatures that are requested by a Security Compliance Manager client before it accepts any collectors:

- IBM (origin) certificate (IBM)

  The IBM collector certificate is included with Security Compliance Manager. This certificate is used by both the client and the server to verify that collectors were provided as part of an official IBM product. The IBM private key is not supplied with the product. The IBM certificate prevents unauthorized third-party or malicious collectors from being used. Alternatively, you can use your own certificate for signing collectors.

- Collector authorization certificate

  The authorization root certificate is generated at installation time and protected by the server password. It is used to create authorization certificates (AC). Authorization certificates are used to digitally sign collectors that can be registered on the server. Clients use certificates created with the authorization root certificate to verify that the collectors they receive were sent from the server.

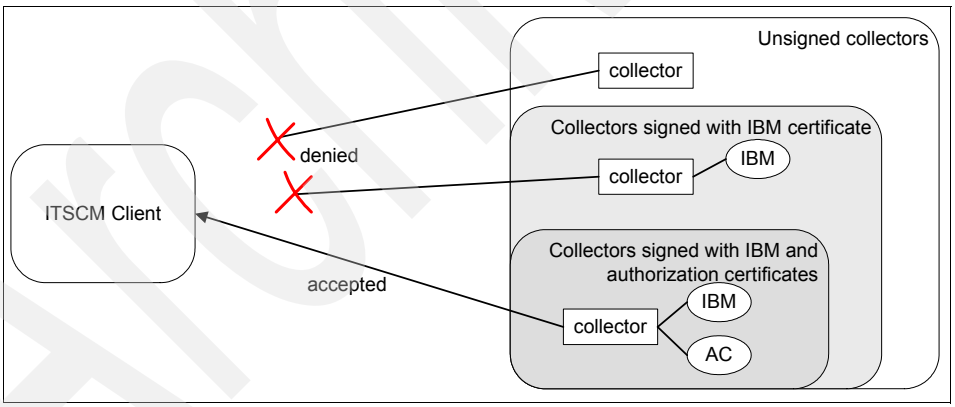


Figure 30-3 Required collector certificate

Proxy relay

The IBM Tivoli Security Compliance Manager proxy relay provides a solution for the scenario of a server separated from destination clients by one or more intermediary networks because of firewall policies or address space concerns. The goal of the proxy relay is to permit the server to successfully connect to and communicate with each destination client system.

Figure 30-4 illustrates that any Security Compliance Manager client may be used as a proxy relay if a special collector called com.ibm.jac.server.JACProxy.jar is added to the Security Compliance Manager client using the Security Compliance Manager administration tools.

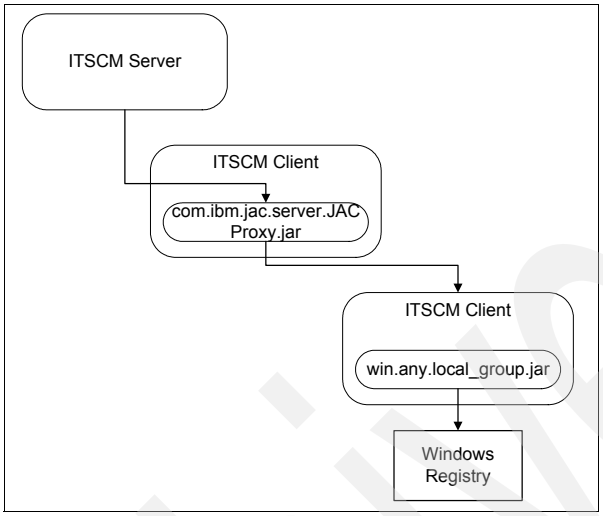


Figure 30-4 Security Compliance Manager client configured as proxy relay

Securing the proxy relay system

The proxy collector is a special collector that permits a client to act as an intermediary between the server and other clients. Because the proxy relay can also be used to bypass a site's security, the proxy relay must possess a method to prevent abuse. The proxy relay enforces a security policy through the use of configurable access control lists (ACLs). An access control list is a security method that uses a set of rules to determine which resources can be accessed by whom and from where. The proxy relay uses two ACLs, one to regulate incoming traffic, and one to regulate outgoing traffic.

30.2.2 Compliance evaluation components

Security Compliance Manager compliance evaluation components extract the data collected, analyze the data for non-compliance, and provide the input for reports in order to reveal adherence to internal and industry-standard security policies.

Security Compliance Manager policy

A Security Compliance Manager policy consists of one or more specially written SQL queries that are used to reveal compliance or violation of system security

requirements. Each SQL query is called a compliance query. A compliance query extracts, from one or more tables, data collected by the collectors, analyzes that data, and then returns the list of clients that are in violation of that specific security requirement.

The results of all compliance queries in a policy can be used to provide a picture, or snapshot, of the level of compliance for all clients under that policy.

Security Compliance Manager snapshot

A snapshot provides the compliance status of all client systems that are associated with a policy. Security Compliance Manager creates a snapshot by running all the compliance queries in a policy against all clients associated with the policy. Users may view the snapshot results using the Security Compliance Manager administration tool, or send the results to one or more e-mail addresses. Snapshot™ administrators can create snapshots on a scheduled basis or can produce snapshots on demand using the administrative utilities. Archiving the results of snapshots on a regular basis is a good practice and can be used to show compliance with both internal security requirements, as well as industry-standard or governmental security and privacy requirements over a period of time.

30.2.3 Compliance report components

There are three types of reports provided by Security Compliance Manager.

- Using the Reports Panel in the admin GUI, you can schedule queries and generate reports.
- You can create snapshot reports (from scheduled snapshots).
- You can use IBM DB2 Alphablox (which includes operational reports and historical reports).

Security Compliance Manager report

Tivoli Security Compliance Manager provides a reporting capability in the administration console. Each report contains the result of a single snapshot and lists the violations and the corresponding client details. A Security Compliance Manager administrator can schedule a report to run on a periodic basis and configure Security Compliance Manager to automatically send the results to a specified e-mail address.

Security Compliance Manager operational reports

Security Compliance Manager provides operational reports for security compliance reporting. Operational reports require DB2 Alphablox, a J2EE-based infrastructure for report delivery. The reporting interface is based on AJAX

technology for better user experience. Figure 30-5 shows the Alphablox Web application set up for Security Compliance Manager reports.

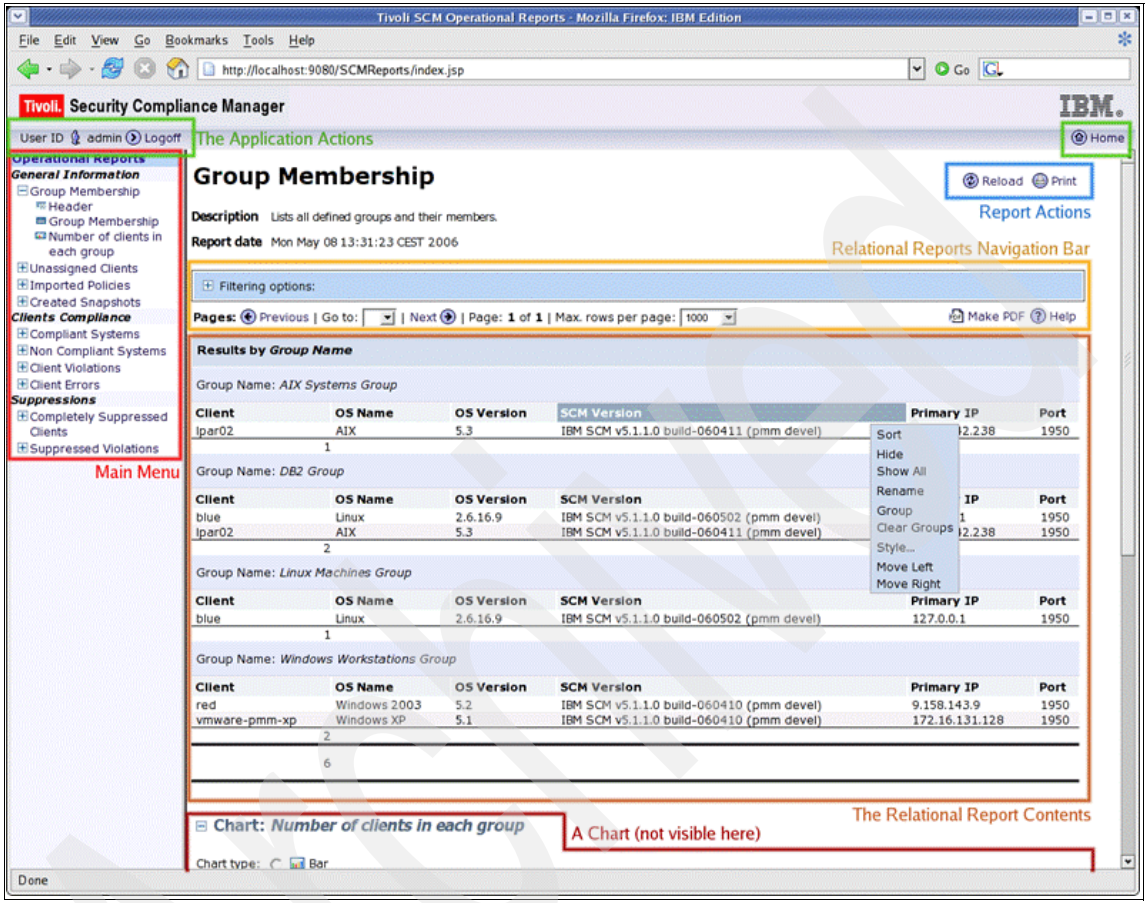


Figure 30-5 Alphablox Web application for Security Compliance Manager

For more information on the installation and configuring of DB2 Alphablox see the following Web address:

`http://publib.boulder.ibm.com/infocenter/ablxhelp/v8r4m0/index.jsp`

The steps to configure Security Compliance Manager to use Alphablox can be found on the release notes of Security Compliance Manager 5.1.1.1 at the following Web address:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itscm.doc_5.1/TivoliSCM5.1.1%20RN_2.0.pdf`

30.2.4 Security Compliance Manager server

The server is Java language-based software that centrally manages all data associated with Tivoli Security Compliance Manager. The Security Compliance Manager server is the central component of a Security Compliance Manager infrastructure and manages compliance report components, compliance evaluation components, and data collection components.

The Security Compliance Manager server stores the data associated with the objects being managed in a centralized DB2 relational database. The server is the only Tivoli Security Compliance Manager component that directly accesses the database. Data can be extracted for system analysis.

Securing the Security Compliance Manager server

The Security Compliance Manager server manages data, which can be an invaluable source of information for all kinds of intruders. The Security Compliance Manager database contains a list of IT systems, IP addresses, user accounts, configuration options, and much more information, which can provide hints for potential starting points for attacks. Tivoli Security Compliance Manager provides the following features to secure the Security Compliance Manager server and its data:

- Secured communication between server and administration console

  The communication between server and administration console is secured by SSL. The administration console verifies the identity of the server based on the server certificate.

- Secured communication between server and client

  The Security Compliance Manager client establishes communication links with the Security Compliance Manager server based on the server's SSL certificate and IP address. Any other communication requests are denied. This ensures that only the authorized server is able to perform configuration requests like collector deployment or schedule changes.

- Protecting the database

  The DB2 database contains valuable information about the IT infrastructure and known vulnerabilities. The node hosting Security Compliance Manager's DB2 database system should be placed in a trusted security zone. Additionally, access to the Security Compliance Manager database should be restricted to the absolute minimum.

Communications between Tivoli Security Compliance Manager components are secured using 128-bit Secure Sockets Layer (SSL) encryption. The cipher suites used are RSA_WITH_RC4_128_SHA, RSA_WITH_RC4_128_MD5, and RSA_WITH_3DES_EDE_CBC_SHA.

Failover support for Security Compliance Manager Server

A new enhancement in Security Compliance Manager version 5.1.1.1 is the failover support for the Security Compliance Manager server. This new feature provides functionality that helps to keep the continuity of clients' data collection despite a Tivoli Security Compliance Manager server outage. The failover support provides a possibility to set up secondary Tivoli Security Compliance Manager servers to handle messages from Tivoli Security Compliance Manager clients in case of a master server failure or to allow for load balancing of the network traffic and server processing.

Attention: In case of master server failure, push clients can communicate with a slave server; however, pull clients can only communicate with the master server. Also snapshot creation, administration console, and command line connections are not available on slave servers.

For information on how to set up a slave server please review the Security Compliance Manager version 5.1.1.1 Release Notes at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itscm.doc_5.1/TivoliSCM5.1.1%20RN_2.0.pdf

30.2.5 Administration components

Administrators and users use the administration components to centrally manage all the other components of the Security Compliance Manager infrastructure. The administration components consist of the Security Compliance Manager administration console and the command line interface (CLI). The following sections describe the administration components.

Administration console

The administration console is the graphical user interface (GUI) used to manage Tivoli Security Compliance Manager servers, clients, collectors, and keystores. The administration console also manages the data collected by the collectors, analyzes that data, and generates reports.

The administration console offers functions to perform the following tasks:

- Manage individual client systems (register and unregister clients)
- Manage client groups (add and remove groups, and add and remove systems to and from groups)
- Manage collectors (install collectors, view status, set values for collector parameters, and customize schedules)

- Manage users (add and remove users, and create and manage user groups and roles)
- Manage proxy relays (define proxy relays and assign routing paths)
- Manage database tables (create delta tables and set maximum data age)
- Manage policies (create, import, and export policies, assign policies to client groups, schedule, run, and view snapshots
- Manage reports (define reports and run reports)
- Define and test SQL database queries
- Manage the server (define authorization keys, view server activity, back up keystores, and manage the database connection)

Command line interface

The command line interface provides an alternative to the administration console and offers a subset of the functions available with the administration console. The command line interface enables the administrator to perform operations on a large number of objects or to automate operations with scripts or batch files. The command line tools are available on all supported platforms.

Detailed information about how to deploy IBM Tivoli Security Compliance Manager is in the IBM Redbooks publication *Deployment Guide Series: IBM Tivoli Security Compliance Manager,* SG24-6450.

31

Tivoli Security Compliance Manager scenarios

Our earlier discussion of Tivoli Security Compliance Manager has been helpful in describing the basic elements of architecture for deployment. At this point, we apply those guidelines to a simple compliance scenario for a fictional organization with a typical set of requirements and we expand the solution to add the remediation and Network Admission Control components as the security requirements of the organization grows.

While host machine configuration and capacity is touched on in this chapter, we deliberately avoid providing much in the way of specifics. This is because without appropriate capacity planning activities, which consider simulated or real loads of the actual application, accurate determinations can be difficult to make. For more detailed technical information, refer to the IBM Redbooks publication *Deployment Guide Series: IBM Tivoli Security Compliance Manager*, SG24-6450.

31.1 Automated security compliance management

This section provides a discussion of the steps to deploy an automated security compliance solution with Tivoli Security Compliance Manager.

31.1.1 Company profile

Medvin, Lasser & Jenkins (ML&J) is a major brokerage firm with headquarters located in the United States.

Recent reports about virus incidents as well as changes in the regulatory requirements (Sarbanes-Oxley Act), have forced the company to rethink the way the existing corporate security policies are executed.

Some of the major concerns are:

- The level of compliance to the security controls are currently checked manually and are susceptible to human error.
- The company is under rising pressure to save costs and the manual work involved in security compliance processes is costly.
- ML&J need to be able to demonstrate that the IT environment is *under control* once the regulatory focus on the operation of IT systems has increased.
- Recent reports show financial losses due to increase of malicious attacks and the company wants to proactively protect themselves against viruses by increasing the frequency of compliance checks.

Technology background

ML&J's current architecture has multiple systems of different vendors using different technologies. ML&J's current IT architecture is depicted in Figure 31-1:

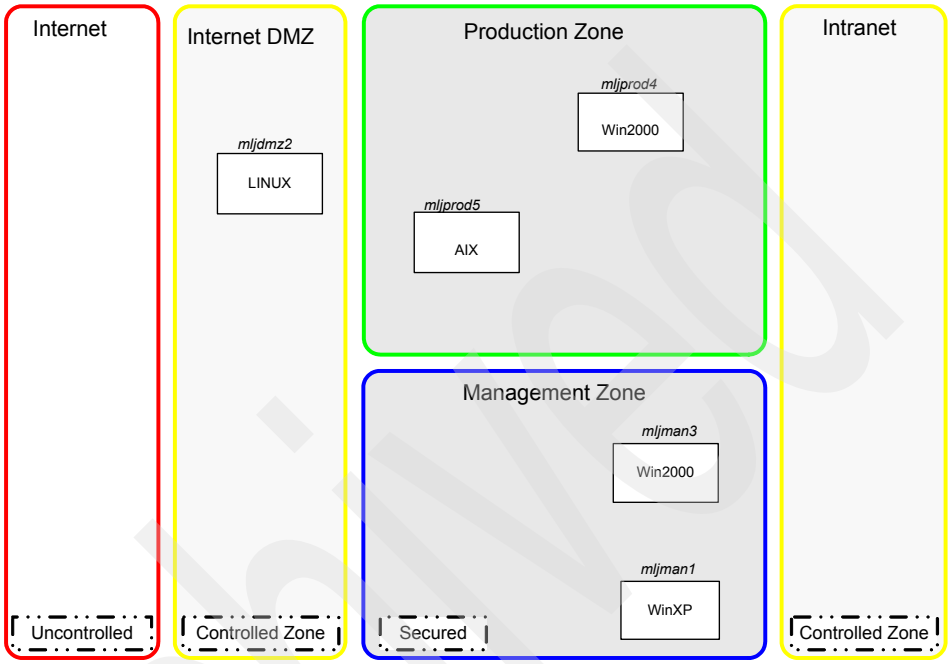


Figure 31-1 ML&J's current IT environment

Initially, ML&J's wants to deploy Tivoli Security Compliance Manager into the following three zones before extending the compliance solution to desktops:

- Internet DMZ
  The Internet DMZ hosts the systems that are responsible to allow only authorized traffic into the ML&J domain.
- Production Network
  The production network is the inner system network containing the business applications.
- Management Network
  The management network hosts management applications required to manage the other networks.

The company wants to deploy the compliance solution without significant impact on their current systems (for example, as few configuration changes as possible, no downtime, and so on).

IT infrastructure

The following components are considered highly critical and therefore included in the scope for compliance checks:

- DB2 databases
- Mail router

Operational plans

Early plans are in the development stage for deployment of antivirus software for all servers. The compliance solution must support and provide the necessary checks to this item.

Solution Administrator

After the deployment of the solution, the IT staff receives training for performing the administrative tasks such as installing Tivoli Security Compliance Manager client components, creating and configuring policies, installing collectors, creating and managing reports.

Tip: Only a small team should have access to the Tivoli Security Compliance Manager Administration GUI because the collected data is considered confidential.

Business requirements

The CIO has provided input about the business drivers for the targeted solution:

- Reduce the window of opportunity (amount of time and number of opportunities) for an attacker to gain unauthorized access to computer systems.
- Automatically check the configuration data obtained from target systems against the security policy.
- Assure and be able to demonstrate the level of compliance of the target systems in a relatively short time.
- Minimize the number of new hardware systems.

Security design objectives

Based on initial discussions and a security workshop, it has been determined that the following key technical requirements exist:

- Collecting the security configuration from the target systems on scheduled intervals.
- Checking the configuration data against a customizable set of values obtained from customers security policy.

- Ensure the integrity of the transmitted data against attacks and prevent any falsification of the components.
- Creating a report on the number of target systems with compliance deviations, reports on the specific deviations per system, and trend reports on the control compliance.

By deploying this solution we are actually delivering *vulnerability management* to the IT staff. They can enforce the security controls and receive reports about the violations, ensure that consistent policies are implemented on servers and manage security risks.

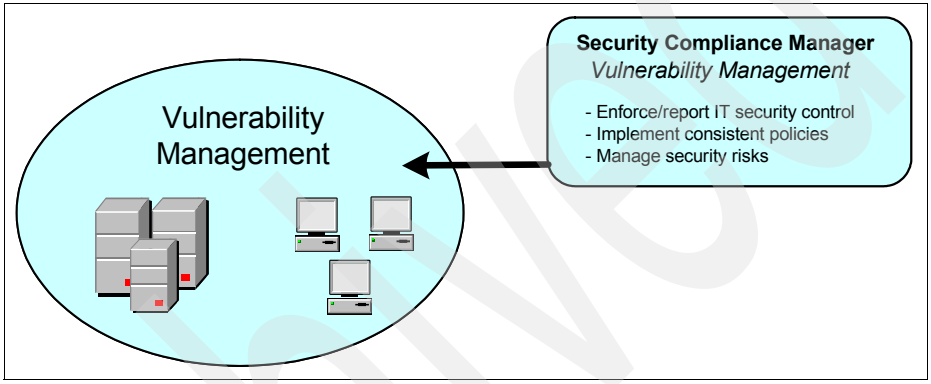


Figure 31-2 Vulnerability management

Requirements analysis

Tivoli Security Compliance Manager is an obvious fit for ML&J's current needs.

To summarize the requirements discussion above, we know the following:

- We need to have the Tivoli Security Compliance Manager covering the servers in three network zones.
- Reuse existing hardware as much as possible.
- Configure Tivoli Security Compliance Manager to collect the security configuration from the target systems on scheduled intervals.
- Ensure that the policy in Tivoli Security Compliance Manager is reflecting ML&J's security policy.
- Tivoli Security Compliance Manager is able to ensure the integrity of the transmitted data by using SSL as a transport level security mechanism and prevent any falsification of the components by cryptographically signing Java components.

- Create reports on the number of target systems with compliance deviations, on the specific deviations per system, and trend reports on the control compliance.

From this, we can address an initial Tivoli Security Compliance Manager architecture for ML&J.

Compliance solution architecture

As we know it today, the diagram in Figure 31-1 on page 927 summarizes the existing IT environment deployed by ML&J. The compliance solution architected for ML&J has to be integrated with this environment.

We will place the Tivoli Security Compliance Manager in the management network because it is the most secure zone and the data collected by the tool is confidential. Security Compliance Manager server will share the same physical machine with the DB2 database.

Tip: It is important to point out that, as the company expands its operations, it may make sense to split the server and the database onto separate machines. This task is easy to implement when the time comes.

The Security Compliance Manager client will be installed on all servers with a supported operating system. See the supported operating systems in *IBM Tivoli Security Compliance Manager Version 5.1 Installation Guide: Client Component*, GC32-1593.

Due to a security policy that prohibits IT resources to directly communicate from the Internet DMZ to the management network, the servers in the DMZ cannot be accessed directly from the management network. Because of this we have to place a Security Compliance Manager proxy relay in the production network to establish communication between the Security Compliance Manager server in the management network and the Security Compliance Manager clients in the DMZ. We have decided to use the mljprod5 server to be the proxy relay system based on the available computing resources.

We can see the final design in the Figure 31-3 on page 931.

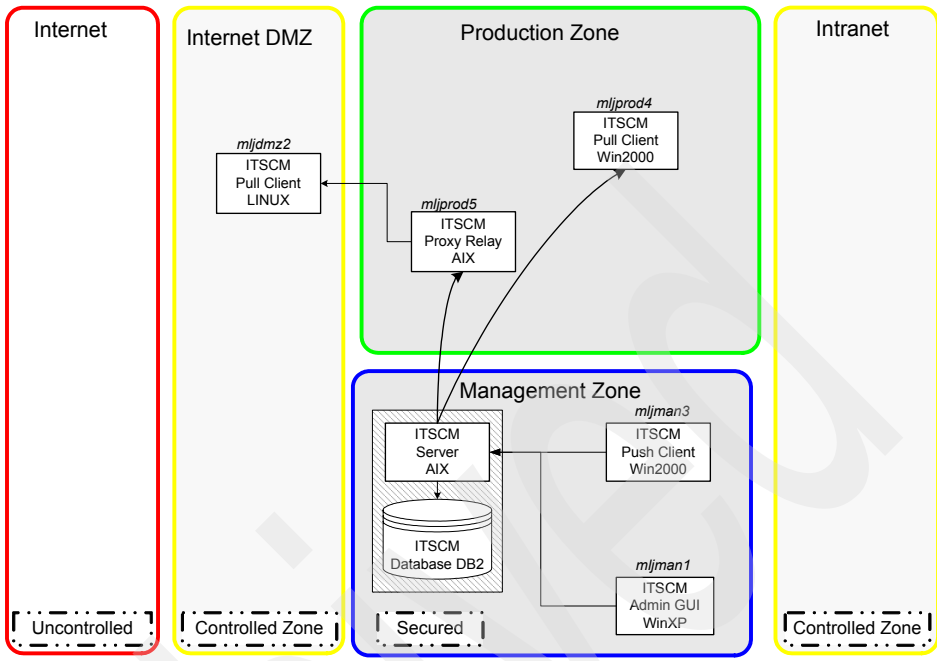


Figure 31-3 Tivoli Security Compliance Manager deployment for ML&J

The next step requires the system administrator for each server to install the Security Compliance Manager client. We must test communication between client and server to ensure that we will be able to collect data from servers.

From the Administration GUI we can now deploy the proxy relay collector to mljprod5 and configure the access control list (ACL) of the proxy relay collector to establish the communication between Security Compliance Manager server and clients in DMZ.

Policies and collectors

After all components have been deployed, we can focus on policies and reports. It is a key task to configure the policies on the Security Compliance Manager server to reflect ML&J's security policy. The DB2 databases and the Mail Router (running on Linux) are considered critical and must receive the policies first.

A previous analysis indicates that the best time to run collectors on ML&J's servers is between 1:00 am and 7:00 am because this is the period with more resources available on server. This information has to be reflected in the policy schedules.

When a policy is added to a group of clients, the collectors that are part of that specific policy are automatically distributed to the Security Compliance Manager client installed on each server member of that group.

At this point, we are ready to schedule snapshots and create reports in the Security Compliance Manager Administration GUI.

Creating reports with DB2 Alphablox

Alphablox provides a Web application that is used for reporting capabilities. End users, such as managers and an audit team, can easily access the reports via a Web site.

The Alphablox server is placed in the same network zone as the Security Compliance Manager server. ML&J plans to consolidate the reporting infrastructure and use the Alphablox server for all kinds of security reports, including security compliance reports, risk management reports, user revalidation reports, and many more.

31.1.2 Summary

In the previous sections we have illustrated the thought process involved in developing a typical Tivoli Security Compliance Manager solution architecture. With this as a base, we can easily extend the architecture to add additional capabilities and capacity, as we will see in later sections.

In the next section we will expand the compliance solution to workstations and add a remediation component.

31.2 Compliance and remediation

In the first phase of the project the major concern was to ensure that the servers are compliant. In the second phase the focus shifts to the desktop compliance with a solution not only to report on policy deviations but to remediate non-compliant machines.

31.2.1 Further evolution

After financial losses due to incidents with malicious software in the intranet, Medvin, Lasser & Jenkins (ML&J) has decided to expand the compliance solution to their desktops and add a remediation solution.

We now face the added design objectives of a larger environment and integration with IBM Tivoli Configuration Manager.

Business requirements

The CIO emphasizes the following business requirements for this expansion:

- ML&J has experienced loss of productivity caused by the introduction of viruses and worms, the spread of which must be stemmed by limiting production network access to systems that comply with the ML&J security policy, such as weekly full-system scans.
- A method to ensure that basic safeguards are employed at the workstation level, such as password quality standards and detection of unauthorized Windows services.
- The utilized method must not heavily consume help desk and system administrator resources.
- Ensure the required hot fixes are installed on all workstations.
- The solution must include a way to remediate non-compliant systems and be built largely upon existing infrastructure to help keep costs at a minimum.
- A minimally intrusive method to institute and enforce emergency change procedures for the company security posture-policy.
- The solution must be able to reverse changes users often perform in local workstations security settings, such as running unauthorized services, that make their workstation inherently less secure.

Security design objectives

The major design objectives of these business requirements target two areas as illustrated in Figure 31-4 on page 934.

- Compliance

  Provide a method to ensure that desktops are configured according to the security policy of the company and have all required hot fixes installed.

- Remediation

  Provide a method to correct the violations found in the compliance phase and install required hotfixes.

By deploying this solution we are delivering *vulnerability management* and *remediation* to the IT staff, adding a new layer of control in the environment.

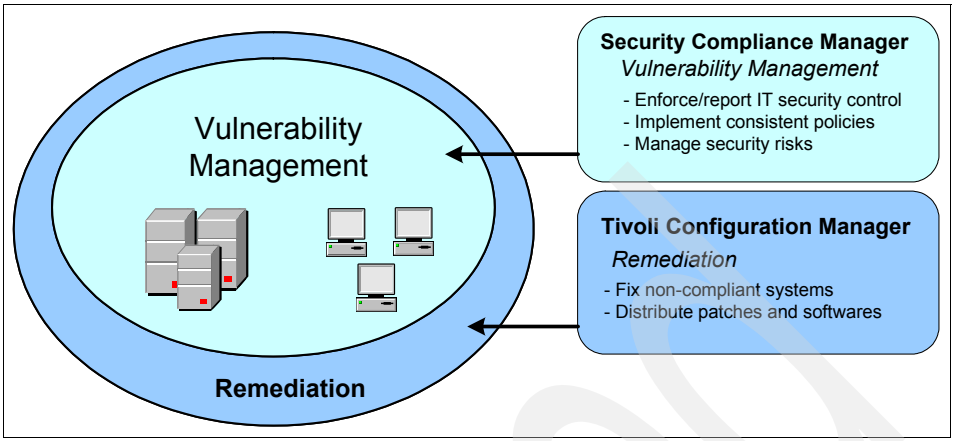


Figure 31-4 Vulnerability management and remediation

31.2.2 Compliance solution architecture

We will place the Tivoli Security Compliance Manager in the management zone and set the firewall rules to permit the communication between server and clients in the specific port used by clients to connect to Security Compliance Manager server. Since this solution is encompassing a lot more clients compared to the previous section, we will install the Security Compliance Manager server on one machine and the database on another to keep the applications from competing for system resources.

Since we have more clients in this scenario and we want to save as many resources as possible on the server, the clients will be installed as *push clients*. The push type requires less resources on the Security Compliance Manager server and once the firewall rules are changed to permit traffic between Security Compliance Manager clients and server, there is no restriction forcing the use of *pull clients*.

The next steps are to install Security Compliance Manager clients, configure policies to reflect the ML&J's security policies, and create and schedule reports.

Figure 31-5 on page 935 illustrates the final compliance architecture.

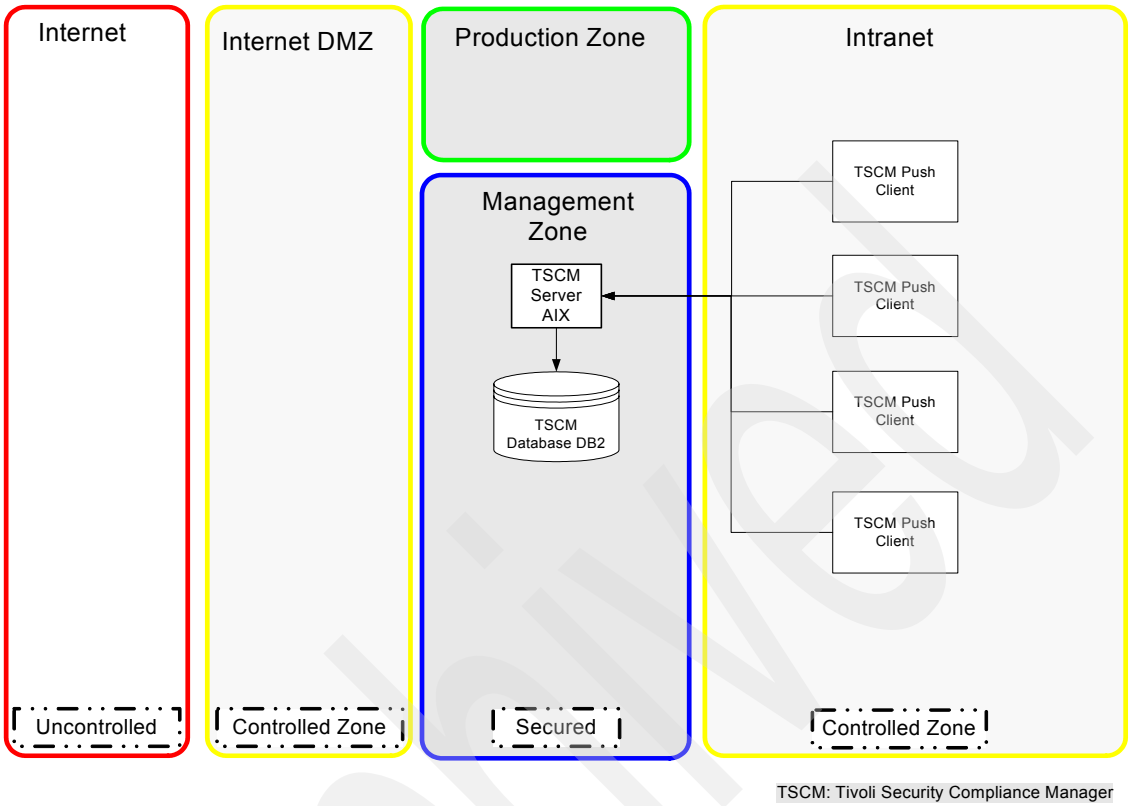


Figure 31-5 Tivoli Security Compliance Manager deployment for ML&J

31.2.3 Tivoli Configuration Manager

IBM Tivoli Configuration Manager has a software distribution capability that enables you to rapidly and efficiently deploy complex mission-critical applications to multiple locations from a central point. After systems have been deployed, the inventory module lets you automatically scan for and collect hardware and software configuration information from computer systems across your enterprise. Tivoli Configuration Manager also has the ability to enforce adherence to your company's policies by changing system configurations as needed to ensure compliance. Tivoli Configuration Manager includes Microsoft software patch automation capabilities designed to save time, lower costs, and improve quality.

Tivoli Configuration Manager can automatically:

- Obtain, package, distribute, and install Microsoft software patches needed by client systems in distributed environments.
- Obtain software patch signature files and distribute them to client machines; scan clients, determine missing patches, package patches, build patch deployment plans, and distribute required patches to clients. Then rescan the client machines to verify successful installation and update the inventory.
- Distribute client/server applications, applications for desktops, mobile devices, and pervasive devices across multi-platform networks.
- Update existing software with newer versions.
- Synchronize software on distributed systems.

The patch automation and distribution software capabilities can help IT managers address the security concern of how to effectively apply patches for Microsoft operating systems, Internet Explorer, Media Player, and keep the desired configuration in each software installed in the machine. In addition to lowering costs through the use of automation to save time and labor, it can also reduce the time needed to close security vulnerabilities.

For detailed information about how to deploy Tivoli Configuration Manager, see *Deployment Guide Series: IBM Tivoli Configuration Manager*, SG24-6454.

31.2.4 Remediation solution architecture

We will now introduce the remediation facility. For the remediation solution we will use the Tivoli Configuration Manager. Figure 31-6 shows the physical location of the components.

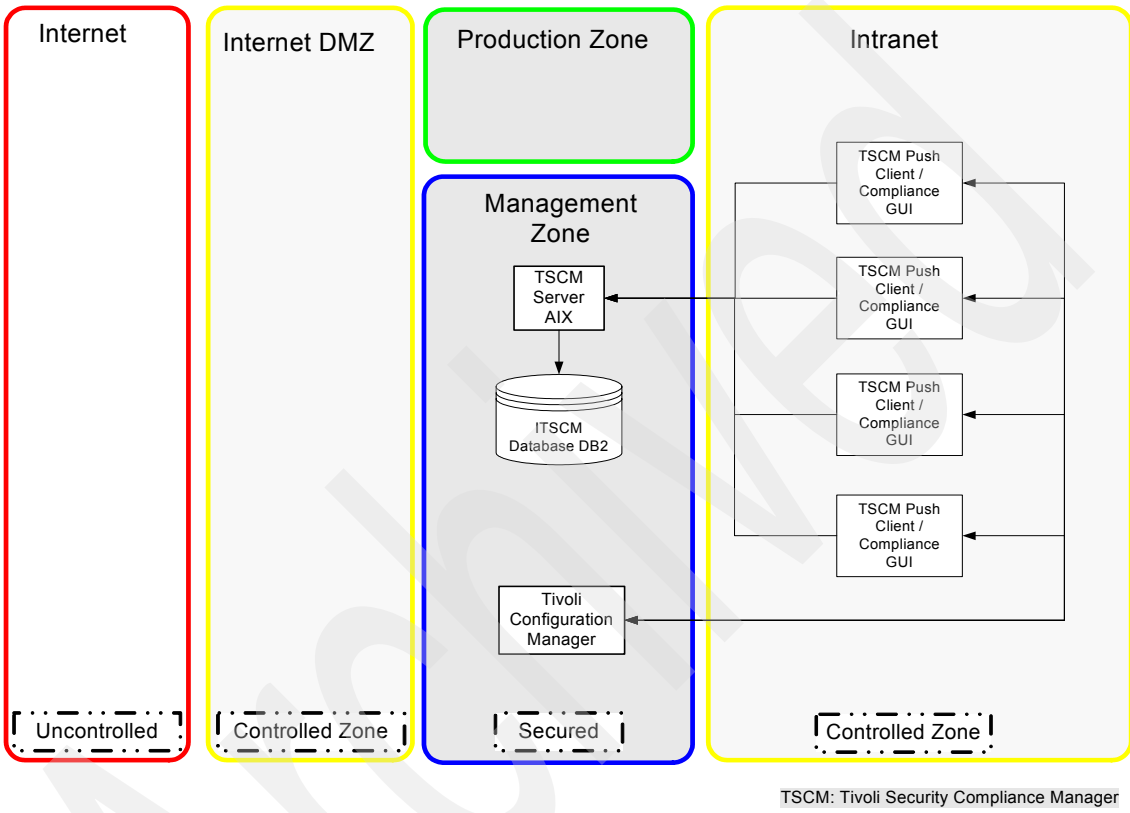


Figure 31-6 Remediation

Figure 31-7 shows the detailed architecture, the components installed on the workstation, and the workflow followed by Tivoli Security Compliance Manager and Tivoli Configuration Manager to ensure compliance and remediation.

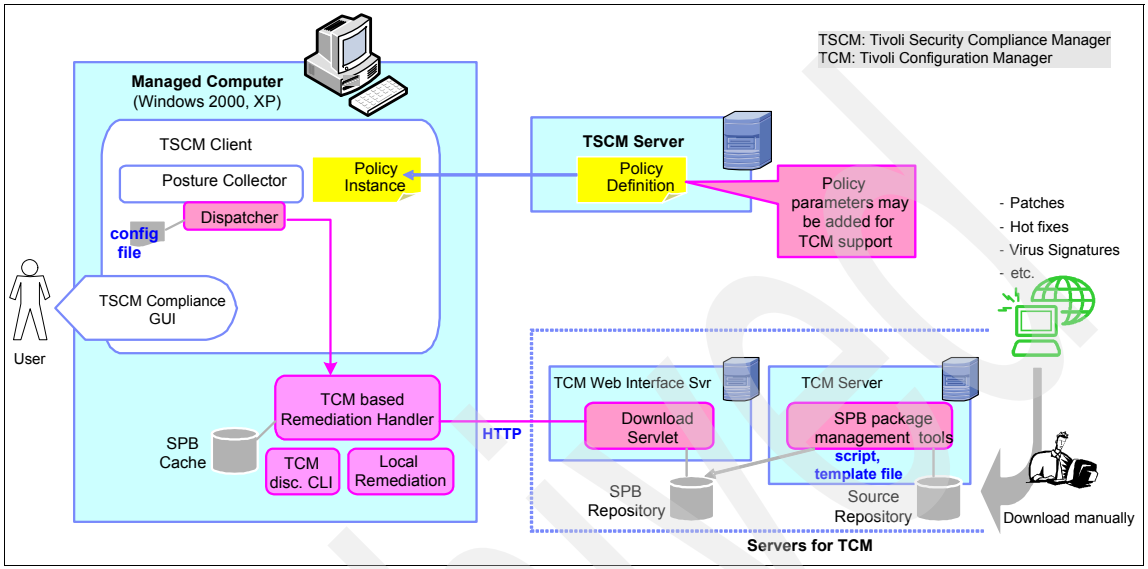


Figure 31-7 Detailed compliance and remediation architecture

The Security Compliance Manager Compliance GUI enables the users to take the necessary actions, such as installing the required hot fix, to bring the workstation back into compliance.

In the Security Compliance Manager Compliance GUI the user can see non-compliant items in their system and then press a button in the remediation handler to fix the problem. This action informs the Tivoli Configuration Manager Based Remediation Handler about the configurations the users want to fix or, when the problem is a missing hot fix, which packages must be automatically copied and installed from the Tivoli Configuration Manager.

Tivoli Security Compliance Manager can be configured to warn the non-compliant workstations after a period of time, reminding users on their actual status and proposing remediation. Security Compliance Manager can also centrally be used to create reports on workstation compliance.

The remediation solution empowers users to correct compliance violations without contacting the help desk or system administrators. It also allows the users to install required hot fixes, the major requirement for this solution.

Compliance enforcement

The Security Compliance Manager Compliance GUI cannot enforce the users to correct the violations found in their systems; however, the IT administrator can use the Tivoli Configuration Manager to enforce the compliance.

The administrator defines a workstation reference model, Tivoli Configuration Manager determines the state of the workstations and automatically generates an activity plan to install or change what is required on each workstation. The activity plan containing those activities that are needed to maintain the preferred configuration of the target. Once the activity plan is created, the administrator submits it for running.

This activity can be scheduled to run at a desired frequency, in order to periodically enforce the compliance of all workstations.

31.2.5 Summary

In this section we described how to expand the solution we have introduced in the previous section and add a remediation component.

This solution can also be deployed using Tivoli Provisioning Manager. For further details on the deployment of the remediation solution with Tivoli Provisioning Manager, see the IBM Redbooks publication *Building a Network Access Control Solution with IBM Tivoli and Cisco Systems*, SG24-6678.

Now that we have the compliance and remediation solution for workstations, in the next section we will enhance security by adding the Network Admission Control component.

31.3 Compliance, remediation, and Network Admission Control scenario

In this chapter we continue the discussion from the previous section with our customer ML&J. In the current project, ML&J wants to integrate their existing compliance and remediation infrastructure with a Network Admission Control solution based on workstation posture-compliance status information.

31.3.1 Further evolution

In order to avoid malicious software and security exposure, ML&J has decided to deny access to the network to non-compliant workstations.

They decided to integrate the current compliance and remediation solution with the Network Admission Control to ensure only compliant workstations can access the intranet zone.

Business requirements

The CIO stated the following business requirements:

- Mobile worker remote access must be maintained; at the same time, increased controls must be put in place to reduce risks to the corporate infrastructure.
- Visitors must have access only to Internet, they cannot access the ML&J's intranet zone.

We find that the following pain points are the requirement drivers:

- Mobile workers present a challenge for the IT staff because of a general lack of ability to ensure that company computer image and update policies are followed.

  Mobile users often move back and forth from client-networks to the ML&J network, thereby increasing the exposure risk.
- Visitors are not being automatically identified and moved out of the intranet zone.
- Locating and isolating non-compliant systems consumes time and resources.

As we examine the business requirements and the pain points, we find that they can be condensed into two simple functional requirements. The first functional requirement is the restriction of network access for non-compliant workstations. The second functional requirement provides a means of remediation for eligible workstations.

Utilizing the existing Tivoli Security Compliance Manager server and Tivoli Configuration Manager infrastructures minimizes incremental equipment and training costs. Note that the Network Admission Control methodology is being deployed only to workstations.

Note: You might not want to extend this solution to servers, once that moving servers out of their network can cause productivity losses.

Security design objectives

It has been determined that the following technical requirements exist:

- Provide a report to users to be aware of which items in their systems are non-compliant.
- Deny access to the intranet zone to non-compliant users and move them to a quarantine network.
- Provide in the quarantine network the tools and instructions to fix the non-compliant items and bring the workstation back to compliance.
- Automatically allow the compliant systems access to the intranet zone.

As this is a major operational shift, the introduction of Network Admission Control technology will not be transparent to the user. Therefore, the security goal is to provide high-quality security without unnecessarily inconveniencing users.

After achieving these objectives we will have a complete solution in place that addresses vulnerability management, remediation, and network access control as we can see in Figure 31-8.

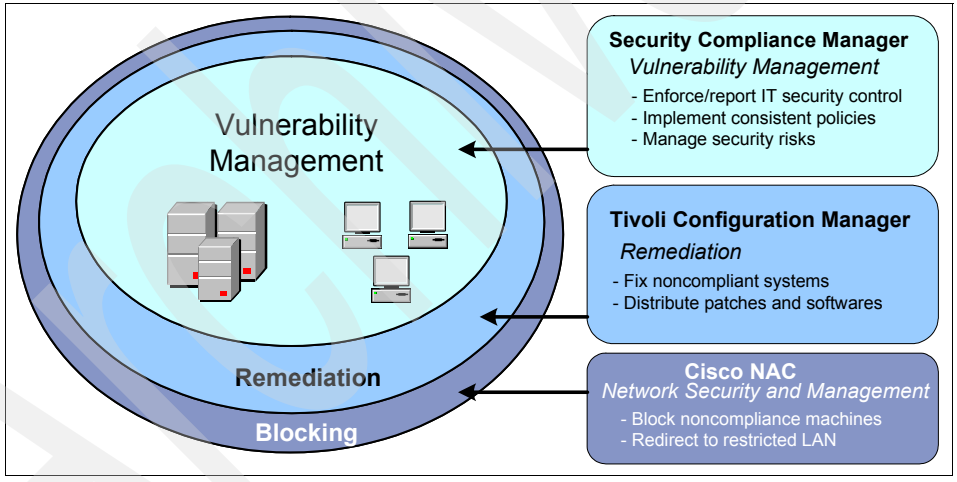


Figure 31-8 Integrated vulnerability management

31.3.2 Solution architecture

A Network Access Control (NAC) is an industry-wide collaboration sponsored by Cisco Systems[1]. An NAC implementation requires a multivendor collection of physical and logical components.

[1] Refer to Cisco Web site for the latest list of supported hardware and corresponding software for the NAC solution at http://www.cisco.com/go/nac

As referenced by Figure 31-9, the major Cisco components include a client-side Cisco Trust Agent, a Cisco Network Access Device (NAD) running an NAC-enabled version of Cisco's IOS, and a Cisco Secure Access Control Server (ACS) running Version 3.3 or later. The major IBM components of the integrated solution include the Tivoli Security Compliance Manager client/server component and the Tivoli Configuration Manager or Tivoli Provisioning Manager remediation client/server code.

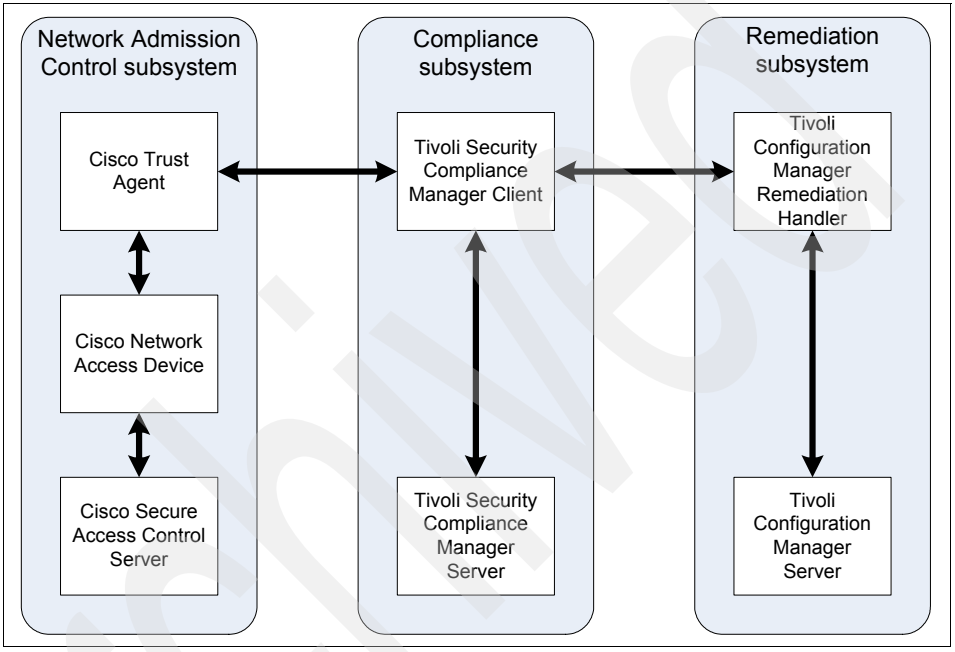


Figure 31-9 Component subsystems - Total solution

In Figure 31-9 we can see that the total solution is comprised of three major subsystems: Network Admission Control, Compliance, and Remediation.

ML&J already have the Compliance and Remediation subsystem in place and configured. In this section we focus on extending the infrastructure to allow for posture policy checks at the workstation level adding the Network Admission Control subsystem.

In logical terms, the Network Admission Control and Compliance subsystem can be spanned into a *network admission policy*. This is comprised of the establishment and enforcement of the compliance criteria.

We need to use a policy in Tivoli Security Compliance Manager server that contains the posture collectors that are used to make client-side compliance

decisions. This policy is imported into the Tivoli Security Compliance Manager environment and modified to meet ML&J's requirements.

After configuration of the policy in Tivoli Security Compliance Manager we must configure the Cisco Secure Access Control Server policy to use the Tivoli Security Compliance Manager agent as a mandatory credential type in the NAC database.

NAC database An NAC database instance consists of a set of *mandatory credential types* and a set of policies.

Mandatory credential types
Cisco Secure ACS uses mandatory credential types to determine whether an NAC database instance should be used to evaluate a posture validation request. If the request includes each of the specified credential types, then Cisco Secure ACS uses the NAC database to evaluate the request.

In our scenario, we list the Cisco Trust Agent and the Tivoli Security Compliance Manager client as our mandatory credential types. From the Cisco Trust Agent credential we extract the operating system type. Thus in all, three pieces of information are used to make the access decision:

- The operation system type
- The Tivoli Security Compliance Manager policy version
- The Tivoli Security Compliance Manager posture policy violation count

When the Tivoli Security Compliance Manager posture policy indicates the presence of violations, the workstation will be logically moved to a quarantine network where quarantined client workstations can only connect to a remediation server.

After the necessary corrections have been made, the workstation is automatically allowed access the intranet zone again.

The flow is summarized in Figure 31-10 on page 944 and explained in the list that follows.

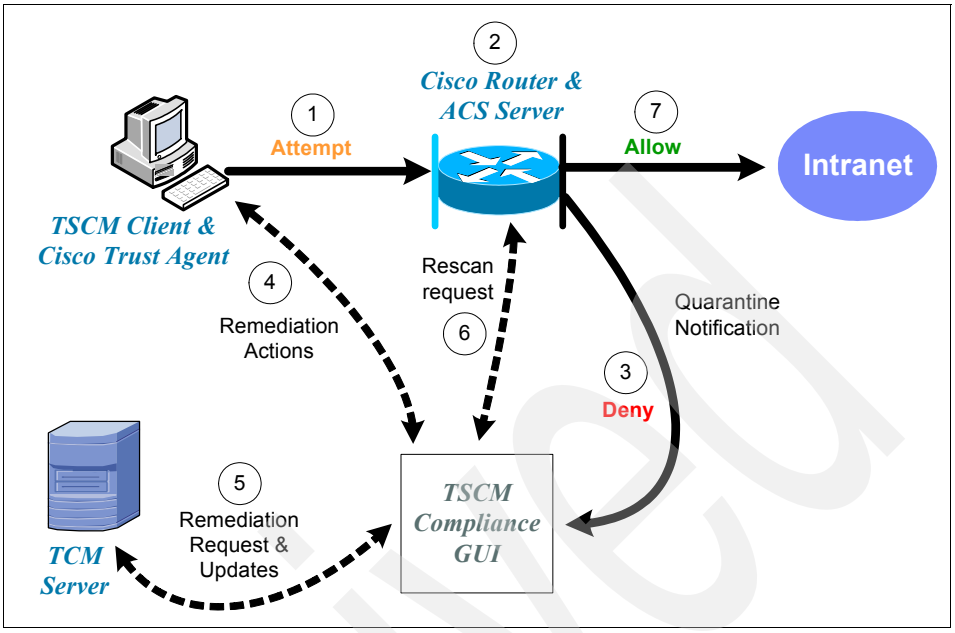


Figure 31-10 Network Access Control and remediation solution

1. The workstation attempts to access the intranet zone.
2. Cisco ACS communicates with Cisco Trust Agent installed on the workstation and challenges the workstation for compliance posture.
   The Security Compliance Manager client determines that there are policy violations on the local device.
3. Access is denied and the workstation is sent to a quarantine network to fix non-compliance.
4. User requests remediation actions from the Security Compliance Manager Compliance GUI.
5. Security Compliance Manager Compliance GUI initiates remediation and updates from Tivoli Configuration Manager server.
6. Security Compliance Manager Compliance GUI requests a new scan from Cisco ACS.
7. The workstation is compliant, the access to intranet is allowed.

In the situation of a visitor workstation, where there is no Cisco Trust Agent to answer to Cisco ACS or a Cisco Trust Agent with a different certificate not recognized by the Cisco ACS in ML&J's environment, the system can be automatically moved to a network that grant access only to an Internet zone.

This guarantees that no visitor can access the intranet zone, with potential confidential content.

31.3.3 Summary

In this section we described how the business objectives are combined with the pain points to drive a set of functional requirements.

Compliance-based Network Admission Control is an emerging technology that brings with it a huge paradigm shift in network security management.

For further details in the deployment of the Network Admission Control and remediation solution, see the IBM Redbooks Publication *Building a Network Access Control Solution with IBM Tivoli and Cisco Systems*, SG24-6678.

A

Method for Architecting Secure Solutions

This appendix introduces a new Method for Architecting Secure Solutions (MASS) that will be used by IBM Global Service employees in future security architecture engagements. It helps understand and categorize security-related problems and discussions in today's e-business-driven enterprise IT infrastructures. This discussion was originally posted in a special edition of the IBM Systems Journal on *End-to-End Security*, Vol. 40, No. 3[1]. We also present an example of using MASS in "Global MASS: An example" on page 979.

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

[1] Copyright 2001 International Business Machines Corporation. Reprinted with permission from IBM Systems Journal, Vol. 40, No. 3.

security functions within a structured design process in order to facilitate greater trust in the operation of resulting IT solutions.

Trust is the measure of confidence that can be placed on the predictable occurrence of an anticipated event or an expected outcome of a process or activity.

For business activities that rely on IT, trust is dependent on both the nature of the agreement between the participants and the correct and reliable operation of the IT solution. The reliance on computerized processes for personal, business, governmental, and legal functions is evolving into a dependency and a presumption (not to be confused with trust) that the processes, and the IT systems within which they execute, will function without flaw. It is reasonable to expect that legal findings relative to the correct and reliable operation of IT solutions will be the basis for whether one party is liable for the damages suffered by another party as a result of a computerized operation.

Trustworthiness of IT solutions can be affected by many factors found throughout the lifecycle of solution definition, design, deployment, and operation. The trustworthiness of design of IT solutions can be affected by the clarity and completeness with which the requirements are expressed by stakeholders and interpreted by solution designers. The trustworthiness of operation of IT solutions can be affected by the trustworthiness of the components and processes upon which they are built, the accuracy with which the design is implemented, and the way in which the resulting computing systems are operated and maintained. The trustworthiness of operational IT solutions can also be affected by the environments in which the solutions are positioned, by individuals who access them, and by events that occur during their operational lifetime.

Given that IT components will most likely continue to have flaws, that unexpected events will most likely occur, and that individuals will most likely continue to seek to interfere with the operation of computing solutions and the environmental infrastructure upon which the solutions rely, what can be done to instill a sufficient measure of confidence (that is, trust) in the correct and reliable operation of a given information technology solution?

One realistic expectation is that designers and integrators of IT solutions will enlist all reasonable measures to effect the correct and reliable operation of IT solutions throughout the design, development, and deployment phases of the solution lifecycle.

While the responsibility for considering all reasonable measures is shared among all individuals involved in the design, development, and deployment of every IT solution, the role of anticipating the perils that the IT solution may face, and ensuring that the business risks of IT solution operation are mitigated, is generally the focus of IT security professionals.

Information technology security is a discipline that until recently was centered within the military, national security organizations, and the banking industry. With the growth of the Internet as a core networking and cooperative computing infrastructure, the need for, and the value of, IT security expertise has increased dramatically. The position of today's security architect closely parallels the role of the network manager or operator of the early 1980s. The similarities include the need to meet high expectations and service levels, a limited set of tools and techniques, low visibility of the electronic activities within the operational environment, plus the challenge of timely recognition and response to events and peril. In the mid-1980s, the development of a systems management discipline provided a focus, a method, and a tool set for standardized approaches to system-wide design, operation, and management.

To date, the application of IT security countermeasures is generally limited to addressing specific vulnerabilities, such as applying network and systems management processes, hardening operating systems for publicly available servers, applying and monitoring intrusion detection systems, configuring and operating digital certificate servers, and installing and configuring firewalls.

Based on the evolution of destructive computer codes and viruses, the repeated breaches of sensitive computer systems, and recurring incidents of compromise of private information stored on networked computing systems, it is reasonable to conclude that the effectiveness of security measures in computing solutions needs to be improved. Recently, security experts from government and industry expressed the need for a more comprehensive approach to describing security requirements and designing secure solutions.

This appendix documents the findings and recommendations of a project for which the initial objective was to develop training materials for a recently defined technical discipline, within IBM Global Services, for security architects. During the project, early attempts to organize and present the prior art dealing with information technology security produced incomplete and unsatisfactory results, leading to the conclusion that a more fundamental analysis was needed. The refocused analysis produced a thought-provoking proposal for articulating, documenting, and synthesizing security within information technology solutions.

Although the project objectives were met, the by-products are different from those first envisioned. The observations and conclusions from the project are summarized within this appendix, including an examination of the basic motivations for implementing security, a review and recategorization of commonly invoked security standards, an analysis of the fundamental elements of security architecture and its design, and some first attempts to render architectural representations.

Problem statement

A systematic approach for applying security throughout information technology solutions is necessary in order to ensure that all reasonable measures are considered by designers, and that the resulting computing systems will function and can be operated in a correct and reliable manner.

In IBM Global Services, the requirement for a method for designing secure solutions is driven from several perspectives:

- There is a need to grow the community of IT architects with a shared security focus.
- There is a need to create synergy among the several technical disciplines within the IT architect profession relative to security issues.
- There is a need to develop consistent designs, because many businesses and organizations have similar security and privacy compliance requirements based on statute, regulation, and industry affiliation, and many enterprises are multinational, with geographically diverse installations operating under similar security policies and practices.

To be effective, the resulting method should use existing security paradigms, integrate with other information technology architectures, and work with today's technologies.

A logical and systematic technique for designing secure solutions has potential value beyond IBM Global Services:

- To individuals, by fostering trust within computing environments that otherwise would be suspect.
- To information technology professionals, by promoting rigor within an emerging discipline of computing science.
- To enterprises, by providing a technical standard with which the effectiveness of information technology designs, and designers, can be evaluated.

Analysis

Information technology architects rely on a wide range of techniques, tools, and reference materials in the solution design process. The results of a design activity may include an operational computing system or a set of documents that describe the system to be constructed from one or more viewpoints and at different levels of granularity. The documents provide a visualization of the system architecture.

To arrive at a system architecture, architects may use personal experience, or they may rely upon documented systematic procedures or methods. In addition to methods, architects prefer to prioritize work and employ data collection techniques to define the problem space and the solution space. Reference materials can include a taxonomy of the problem space, a catalog of solution requirements, and documented models, patterns, or integrated solution frameworks. In general, as the definition of a given problem space matures, the taxonomy of the solution requirements stabilizes. This leads to well-defined reference models, proven solution frameworks, and mature solution design methods.

IT security architecture fits this model for limited problem spaces such as securing a network perimeter, where a set of solution requirements can be defined. A solution framework can be constructed for an enterprise firewall, and a solution architecture can be documented using known reference models for *demilitarized zones*. (Refer to Chapter 2, “Common security architecture and network models” on page 19.) In general, IT security does not fit this model for the following reasons:

- The security problem space has not stabilized in that the number and type of threats continue to grow and change.
- Existing security solution frameworks take a limited view of the problem space, as with firewalls and network-level security.
- Methods for creating security solution architectures are generally confined to the defined solution frameworks. For ill-defined problem spaces such as IT security, the path to maturity of models and methods requires a different approach.

Security-specific taxonomies, models, and methods

ISO (International Organization for Standardization) 7498-2[6] is a widely referenced document associated with IT security solution design. Its purpose is to extend the applicability of the seven-layer OSI (Open Systems Interconnection) system model to cover secure communication between systems. Section 5 of this document describes a set of security services and mechanisms that could be invoked at the appropriate layer within the OSI system model, in appropriate combinations, to satisfy security policy requirements. Section 8 documents the need for ongoing management of OSI security services and mechanisms to include management of cryptographic functions, network traffic padding, and event handling.

Many security practitioners use the OSI security services (authentication, access control, data confidentiality, data integrity, and nonrepudiation) as the complete taxonomy for the security requirements for IT solutions. However, the preamble of ISO 7498-2 specifically states that “... OSI security is not concerned with

security measures needed in end systems, installations, and organizations, except where these have implications on the choice and position of security services visible in OSI. These latter aspects of security may be standardized but not within the scope of OSI Recommendations."

Security evaluation criteria: Agencies and standards bodies within governments of several nations have developed evaluation criteria for security within computing technology. In the United States, the document has the designation "Trusted Computer System Evaluation Criteria," or TCSEC. The European Commission has published the Information Technology Security Evaluation Criteria, also known as ITSEC, and the Canadian government has published the Canadian Trusted Computer Product Evaluation Criteria, or CTCPEC. In 1996, these initiatives were officially combined into a document known as the Common Criteria, or CC. In 1999, this document was approved as a standard by the International Organization for Standardization. This initiative opens the way to worldwide mutual recognition of product evaluation results.

Common Criteria

Common Criteria provide a taxonomy for evaluating security functionality through a set of functional and assurance requirements. The Common Criteria include 11 functional classes of requirements:

- Security audit
- Communication
- Cryptographic support
- User data protection
- Identification and authentication
- Management of security functions
- Privacy
- Protection of security functions
- Resource utilization
- Component access
- Trusted path or channel

These 11 functional classes are further divided into 66 families, each containing a number of component criteria. There are approximately 130 component criteria currently documented, with the recognition that designers may add additional component criteria to a specific design. There is a formal process for adopting component criteria through the Common Criteria administrative body, which can be found at:

http://csrc.nist.gov/cc/

Governments and industry groups are developing functional descriptions for security hardware and software using the Common Criteria. These documents, known as protection profiles, describe groupings of security functions that are appropriate for a given security component or technology. The underlying motivations for developing protection profiles include incentives to vendors to deliver standard functionality within security products and reduction of risk in information technology procurement. In concert with the work to define protection profiles, manufacturers of security-related computer software and hardware components are creating documentation that explains the security functionality of their products in relation to accepted protection profiles. These documents are called “security targets.” Manufacturers can submit their products and security targets to independently licensed testing facilities for evaluation in order to receive compliance certificates.

Common Criteria: a taxonomy for requirements and solutions

The security requirements defined within the Common Criteria have international support as “best practices.” Common Criteria are intended as a standard for evaluation of security functionality in products. They have limitations in describing end-to-end security; because the functional requirements apply to individual products, their use in a complex IT solution is not intuitive. Protection profiles aid in the description of solution frameworks, although each protection profile is limited in scope to the specification of functions to be found in a single hardware or software product.

Common Criteria: a reference model

The Common Criteria introduce a few architectural constructs: the target of evaluation, or TOE, represents the component under design, and the TOE security functions document, or TSF, represents that portion of the TOE responsible for security. Under Common Criteria, the system or component under consideration is a “black box”; it exhibits some security functionality and some protection mechanisms for the embedded security functions.

Summary of analysis

For well-understood problem spaces, methods document the prior work and provide best practices for future analysis. For changing problem spaces such as IT security, methods can only postulate a consistent frame of reference for practitioners in order to encourage the development of future best practices. With time and experience, the methods and models associated with IT security will mature.

The Common Criteria document has important value to the security community, given its history and acceptance as a standard for security requirements definition, and its linkage to available security technologies through documented

protection profiles and security targets. Common Criteria do not provide all of the guidance and reference materials needed for security design.

To develop an extensible method for designing secure solutions, additional work is required:

1. A system model that is representative of the functional aspects of security within complex solutions.
2. A systematic approach for creating security architectures based on the Common Criteria requirements taxonomy and the corresponding security system model.

System model for security

Eberhardt Rechtin[2] suggests an approach for developing an architecture, differentiating between the *system* (what is built), the *model* (a description of the system to be built), the *system architecture* (the structure of the system), and the *overall architecture* (an inclusive set consisting of the system architecture, its function, the environment within which it will live, and the process used to build and operate it).

For the purposes of this project, the type of IT solutions addressed is consistent with a networked information system (NIS). Furthermore, the overall architecture is represented by the security architecture found within an NIS, and the security architecture is represented by the structure of a security system model. With a generalized system model for security in an NIS environment, architects could create instances of the system model, based upon detailed functional and risk management requirements.

Rechtin outlines the steps for creating a model as follows:

1. Aggregating closely related functions
2. Partitioning or reducing the model into its parts
3. Fitting or integrating components and subsystems together into a functioning system

The security system model will be represented by the aggregation of security functions, expressed in terms of subsystems and how the subsystems interact. The security-related functions within an NIS can be described as a coordinated set of processes that are distributed throughout the computing environment. The notion of distributed security systems, coordinated by design and deployment, meets the intuitive expectation that security within an NIS should be considered

[2] E. Rechtin, *Systems Architecting: Creating and Building Complex Systems*, Prentice Hall, 1991.

pervasive. In an NIS environment, security subsystems must be considered as abstract constructs in order to follow Rechtin's definition.

For this project, Common Criteria were considered to be the description of the complete function of the security system model. The classes and families within the Common Criteria represent an aggregation of requirements; however, after careful review, it was determined that the class and family structures defined within Common Criteria do not lend themselves to be used as part of a taxonomy for pervasive security. The aggregation is more reflective of abstract security themes, such as cryptographic operations and data protection, rather than security in the context of IT operational function. To suit the objective of this project, the Common Criteria functional criteria were re-examined and reaggregated, removing the class and family structures. An analysis of the 130 component-level requirements in relation to their function within an NIS solution suggests a partitioning into five operational categories:

- Audit
- Access control
- Flow control
- Identity and credentials
- Solution integrity

A summary mapping of CC classes to functional categories is provided in Table A-1.

Table A-1 Placing Common Criteria classes in functional categories

Functional category	Common Criteria functional class
Audit	Audit, component protection, and resource utilization
Access control	Data protection, component protection, security management, component access, cryptographic support, identification and authentication, communication, and trusted path/channel
Flow control	Communication, cryptographic support, data protection, component protection, trusted path/channel, and privacy
Identity/credentials	Cryptographic support, data protection, component protection, identification and authentication, component access, security management, and trusted path/channel
Solution integrity	Cryptographic support, data protection, component protection, resource utilization, and security management

While redundancy is apparent at the class level, there is only a small overlap at the family level of the hierarchy defined within Common Criteria and below. Much

of the overlap represents the intersection of function and interdependency among the categories.

Security subsystems

The component-level guidance of Common Criteria documents contains rules, decision criteria, functions, actions, and mechanisms. This structure supports the assertion that the five categories described in Table A-1 on page 955 represent a set of interrelated processes, or subsystems, for security. The notion of a security subsystem has been proposed previously; the authors of *Trust in Cyberspace*[3] described functions within operating system access control components as belonging to a decision subsystem or an enforcement subsystem. The five interrelated security subsystems proposed here and depicted in Figure A-1 expand the operating system-based concept and suggest that function and interdependency of security-related functions, beyond centralized access control, can be modeled as well.

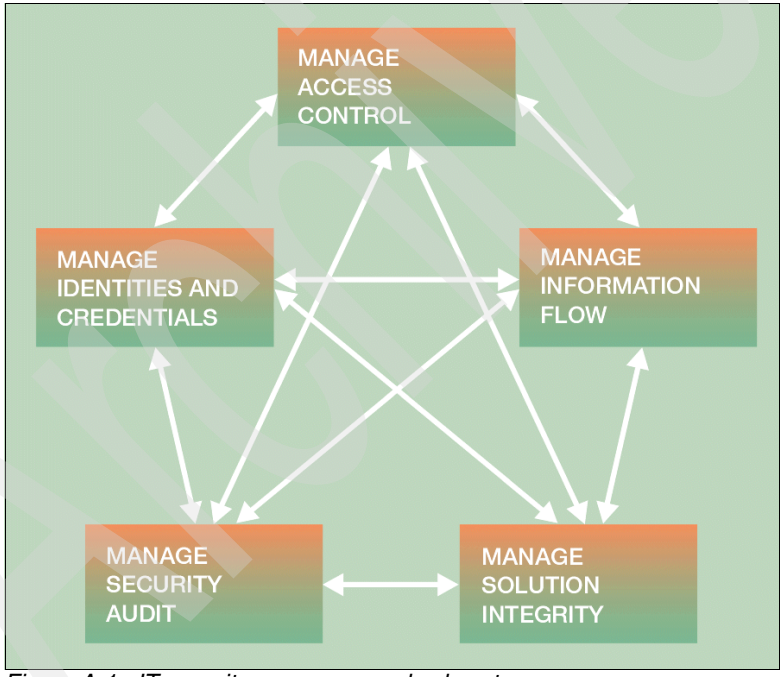


Figure A-1 IT security processes and subsystems

A brief description of each of the five security subsystems, along with further detail of the aggregation of CC component-level criteria within each subsystem,

[3] Committee on Information Systems Trustworthiness, National Research Council, *Trust in Cyberspace*, National Academy Press, 1999.

is now provided. The subsystem diagrams are represented as parts of a closed-loop control system showing the internal processes that each performs, along with its external interfaces. In this representation, each subsystem consists of a managing process with a default idle state and several execution paths that can be invoked either by an asynchronous request signaled by another security subsystem or by a synchronized request from a nonsecurity process. Complementary representations composed of component views and interaction diagrams for the subsystems are being developed.

Security audit subsystem

The purpose of the security audit system in an IT solution is to address the data collection, analysis, and archival requirements of a computing solution in support of meeting the standards of proof required by the IT environment. A security audit subsystem is responsible for capturing, analyzing, reporting, archiving, and retrieving records of events and conditions within a computing solution. This subsystem can be a discrete set of components acting alone or a coordinated set of mechanisms among the several components in the solution. Security audit analysis and reporting can include real-time review, as implemented in intrusion detection components, or after-the-fact review, as associated with forensic analysis in defense of repudiation claims. A security audit subsystem may rely on other security subsystems in order to manage access to audit-related systems, processes, and data; control the integrity and flow of audit information; and manage the privacy of audit data. From Common Criteria, security requirements for an audit subsystem would include:

- Collection of security audit data, including capture of the appropriate data, trusted transfer of audit data, and synchronization of chronologies
- Protection of security audit data, including use of time stamps, signing events, and storage integrity to prevent loss of data
- Analysis of security audit data, including review, anomaly detection, violation analysis, and attack analysis using simple heuristics or complex heuristics
- Alarms for loss thresholds, warning conditions, and critical events

The closed loop process for a security audit subsystem is represented in Figure A-2 on page 958.

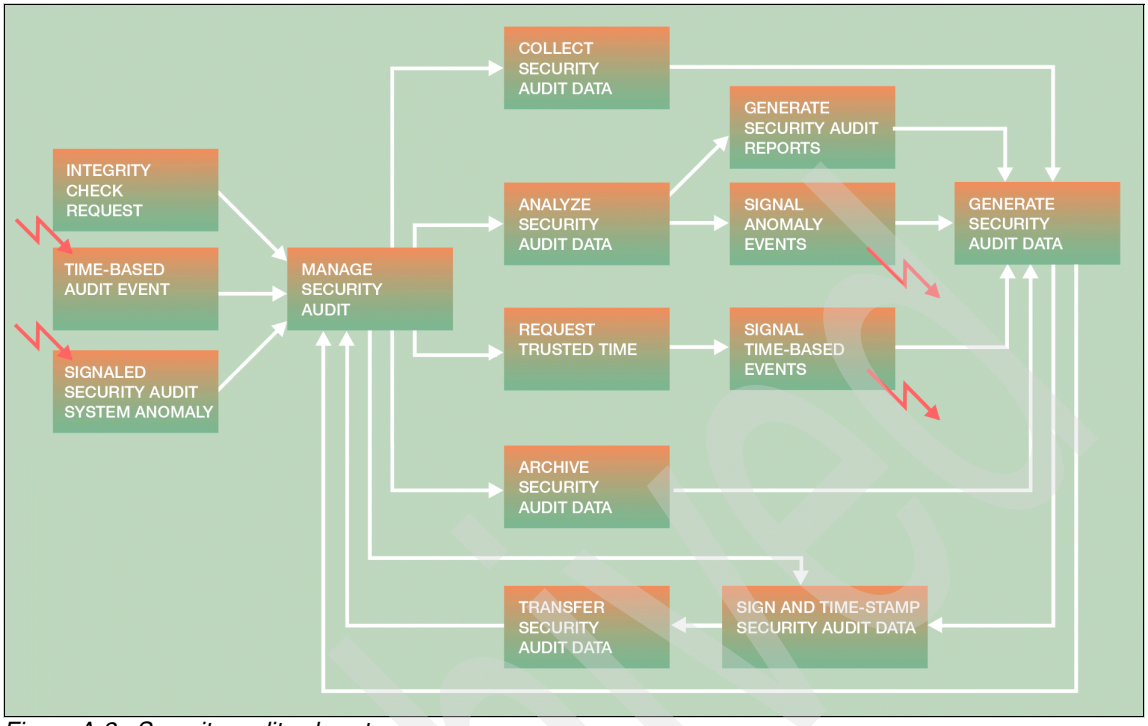


Figure A-2 Security audit subsystem processes

Solution integrity subsystem

The purpose of the solution integrity subsystem in an IT solution is to address the requirement for reliable and correct operation of a computing solution in support of meeting the legal and technical standard for its processes. A solution integrity subsystem can be a discrete set of components or a coordinated set of mechanisms among the several components in the solution. The solution integrity subsystem may rely on the audit subsystem to provide real-time review and alert of attacks, outages, or degraded operations, or after-the-fact reporting in support of capacity and performance analysis. The solution integrity subsystem may also rely on the other subsystems to control access and flow. From Common Criteria, the focus of a solution integrity subsystem could include:

- Integrity and reliability of resources
- Physical protections for data objects, such as cryptographic keys, and physical components, such as cabling, hardware, and so on.
- Continued operations including fault tolerance, failure recovery, and self-testing
- Storage mechanisms: cryptography and hardware security modules

- Accurate time source for time measurement and time stamps
- Prioritization of service via resource allocation or quotas
- Functional isolation using domain separation or a reference monitor
- Alarms and actions when physical or passive attack is detected

Figure A-3 illustrates the closed loop process for a solution integrity subsystem.

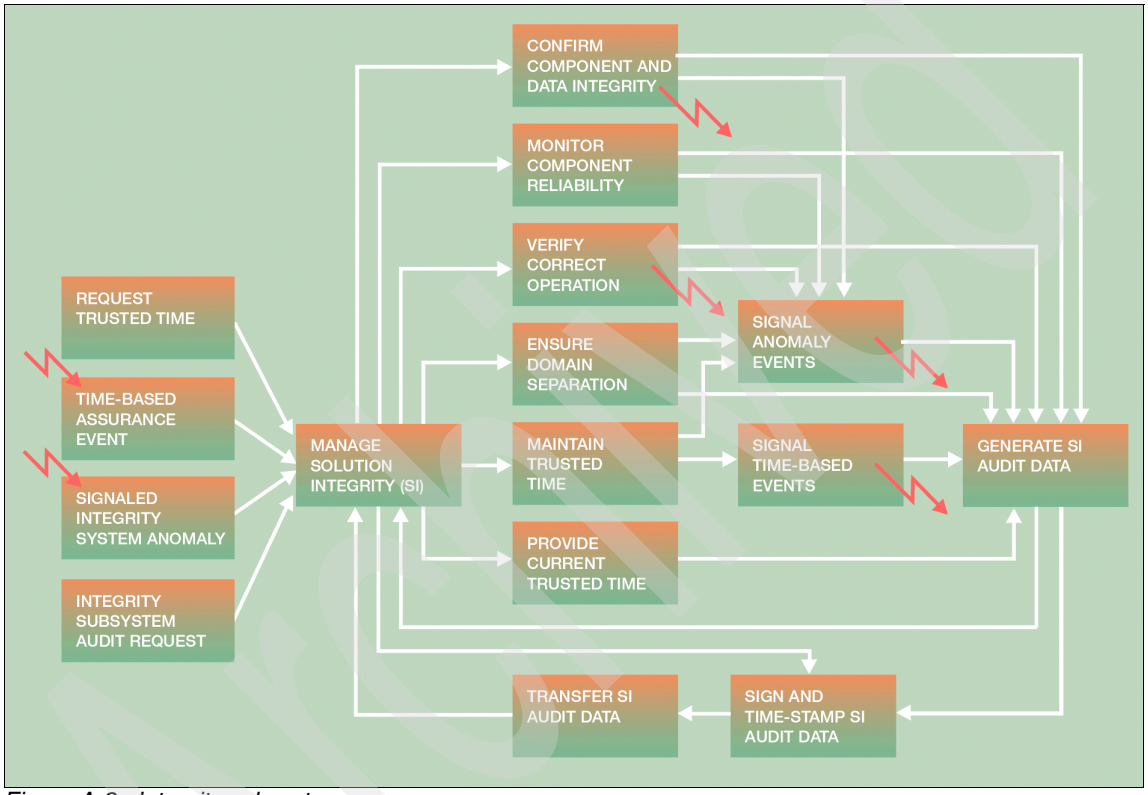


Figure A-3 Integrity subsystem processes

Access control subsystem

The purpose of an access control subsystem in an IT solution is to enforce security policies by gating access to, and execution of, processes and services within a computing solution via identification, authentication, and authorization processes, along with security mechanisms that use credentials and attributes. The credentials and attributes used by the access control subsystem along with the identification and authentication mechanisms are defined by a corresponding credential subsystem. The access control subsystem may feed event information to the audit subsystem, which may provide real-time or forensic analysis of

events. The access control subsystem may take corrective action based on alert notification from the security audit subsystem. From Common Criteria, the functional requirements for an access control subsystem should include:

- Access control enablement
- Access control monitoring and enforcement
- Identification and authentication mechanisms, including verification of secrets, cryptography (encryption and signing), and single-use versus multiple-use authentication mechanisms
- Authorization mechanisms, to include attributes, privileges, and permissions
- Access control mechanisms, to include attribute-based access control on subjects and objects and user-subject binding
- Enforcement mechanisms, including failure handling, bypass prevention, banners, timing and timeout, event capture, and decision and logging components

Figure A-4 illustrates the closed loop process for an access control subsystem.

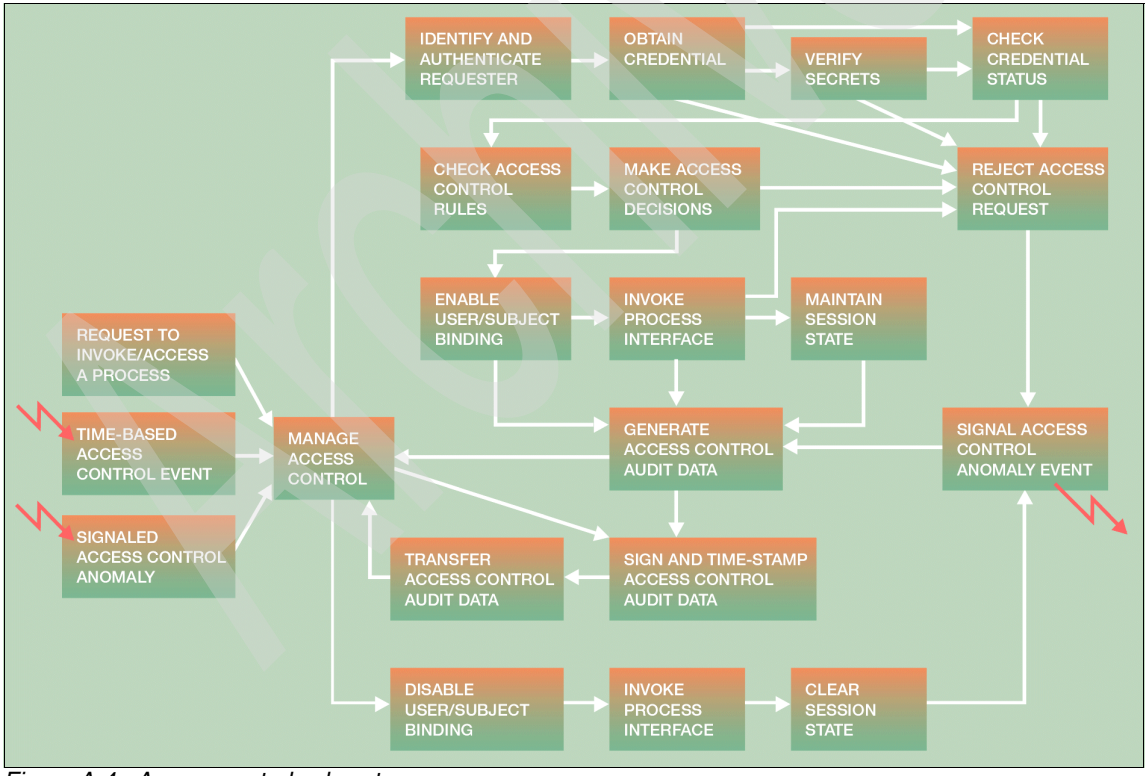


Figure A-4 Access control subsystem processes

Information flow control subsystem

The purpose of an information flow control subsystem in an IT solution is to enforce security policies by gating the flow of information within a computing solution, affecting the visibility of information within a computing solution, and ensuring the integrity of information flowing within a computing solution. The information flow control subsystem may depend on trusted credentials and access control mechanisms.

This subsystem may feed event information to the security audit subsystem, which may provide real-time or forensic analysis of events. The information flow control subsystem may take corrective action based on alert notification from the security audit subsystem. From Common Criteria, an information flow control subsystem may include the following functional requirements:

- Flow permission or prevention
- Flow monitoring and enforcement
- Transfer services and environments: open or trusted channel, open or trusted path, media conversions, manual transfer, and import to or export between domains
- Mechanisms observability: to block cryptography (encryption)
- Storage mechanisms: cryptography and hardware security modules
- Enforcement mechanisms: asset and attribute binding, event capture, decision and logging components, stored data monitoring, rollback, and residual information protection and destruction

Figure A-5 on page 962 illustrates the closed loop process for an information flow control subsystem.

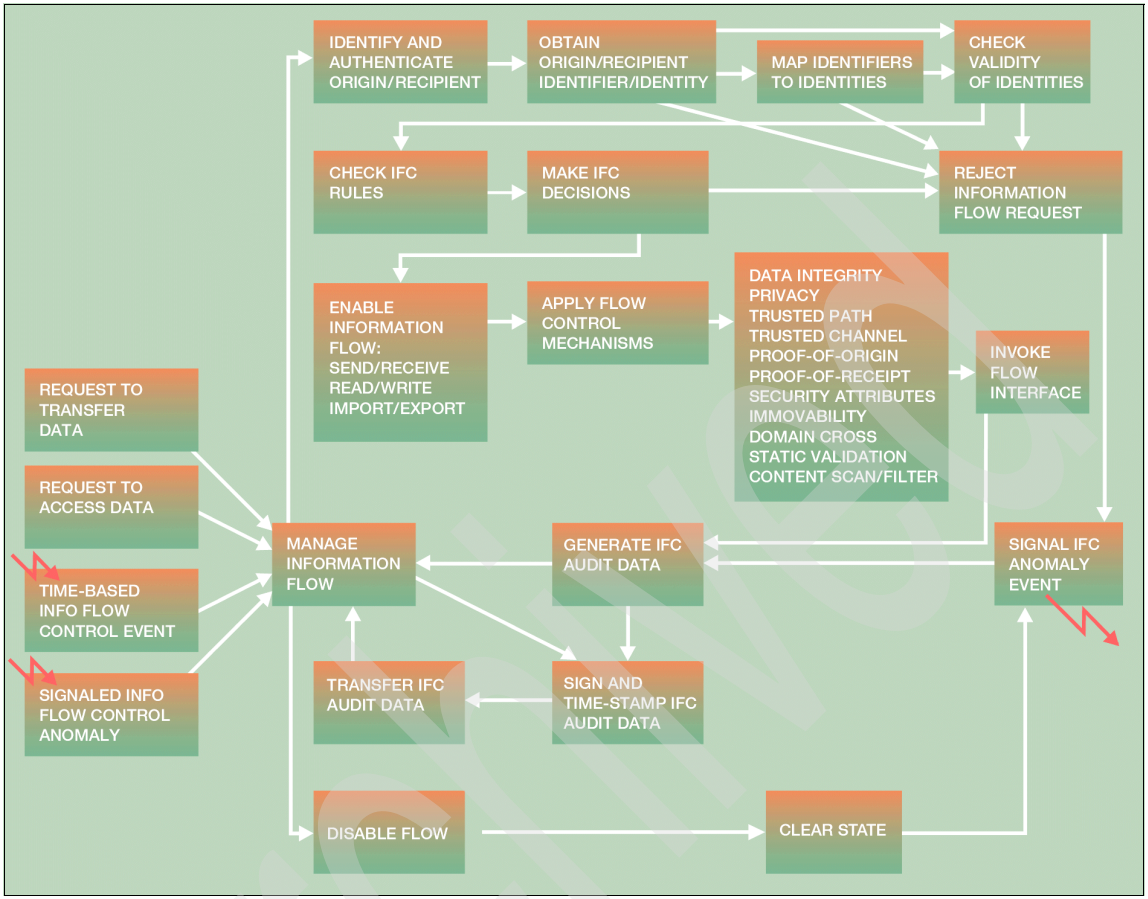


Figure A-5 Information flow control subsystem processes

Identity or credential subsystem

The purpose of a credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms, the processes, and the security subsystems within a computing solution. In some applications, credential systems may be required to adhere to legal criteria for creation and maintenance of trusted identity used within legally binding transactions.

A credential subsystem may rely on other subsystems in order to manage the distribution, integrity, and accuracy of credentials. A credential subsystem has, potentially, a more direct link to operational business activities than the other security subsystems, owing to the fact that enrollment and user support are

integral parts of the control processes it contains. From Common Criteria, a credential subsystem may include the following functional requirements:

- Single-use versus multiple-use mechanisms, either cryptographic or non-cryptographic
- Generation and verification of secrets
- Identities and credentials to be used to protect security flows or business process flows
- Identities and credentials to be used in protection of assets: integrity or non-observability
- Identities and credentials to be used in access control: identification, authentication, and access control for the purpose of user-subject binding
- Credentials to be used for purposes of identity in legally binding transactions
- Timing and duration of identification and authentication
- Lifecycle of credentials
- Anonymity and pseudonymity mechanisms

Figure A-6 on page 964 illustrates the closed loop process for a credential subsystem.

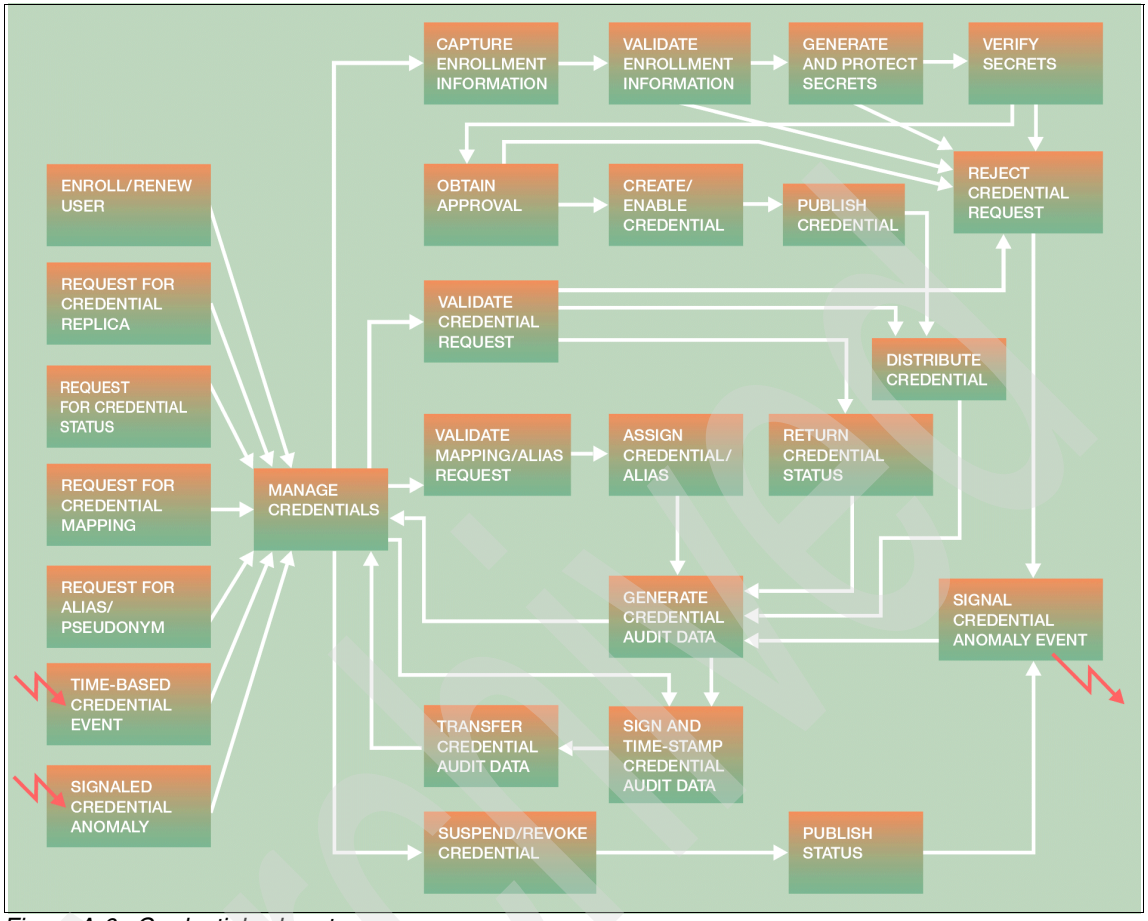


Figure A-6 Credential subsystem processes

Summary of the security system model

This study postulates that the five security subsystems described here exist within every IT solution at the conceptual level, and that the design, integration, and interworking of the services and mechanisms associated with these subsystems represent the security functionality of the solution. This *security system model* needs to be combined with a method for developing the detailed security architecture for a given IT solution.

Developing security architectures

A system architecture has been defined as *the structure of the system to be built*. In this study, the system to be built consists of the security control system found within a networked information system. Figure A-7 represents the solution environment. Here, an e-business computing solution serves information or supports electronic commerce transactions via the Internet. The e-business computing solution is operated by an enterprise and provides services to one or more user communities.

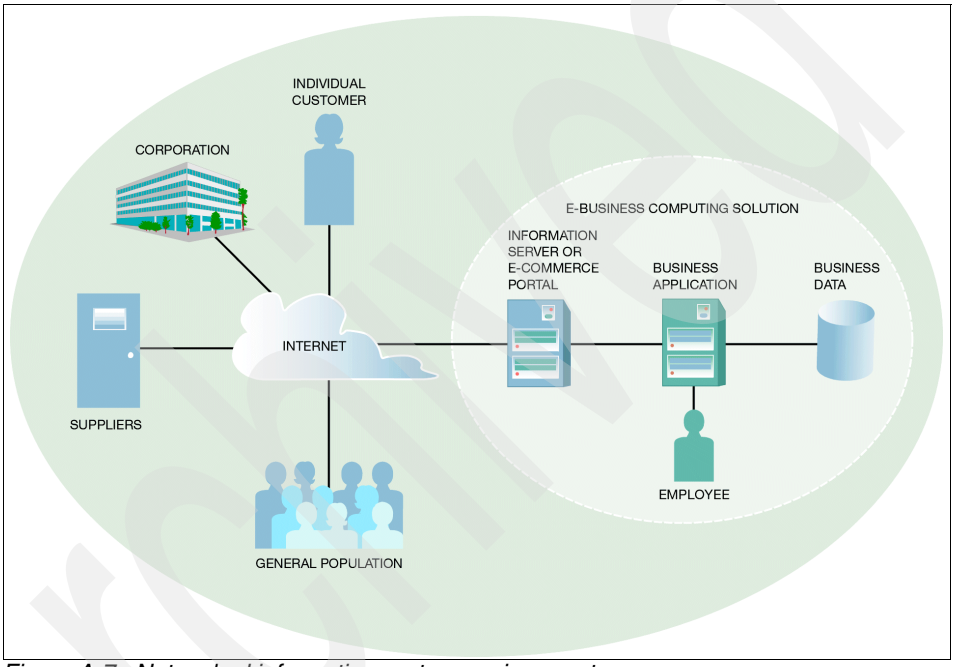


Figure A-7 Networked information system environment

The e-business computing solution can be described as a set of automated business processes supporting the business context that requires security assurances and protections. The design goal is to infuse security into the computing solution and the related IT environment.

From a business perspective, there are two objectives:

- To ensure that the desired IT business process flow yields correct and reliable results
- To ensure that the potential vulnerabilities and exception conditions (that is, perils) within IT business process flows are addressed in ways that are consistent with the risk management objectives

These objectives show the duality of security design: to support and assure normal flows and to identify and account for all illicit flows and anomalous events.

Business process model

Figure A-8 represents IT process flows for a generalized business system. The process flows reflect the events and conditions in which information assets are acted on by processes that are invoked by users, or by processes acting on behalf of users. The left arrow represents the model business flow within a trusted environment, and the right arrow represents a more realistic view of the business flow, where perils exist in the operating environment.

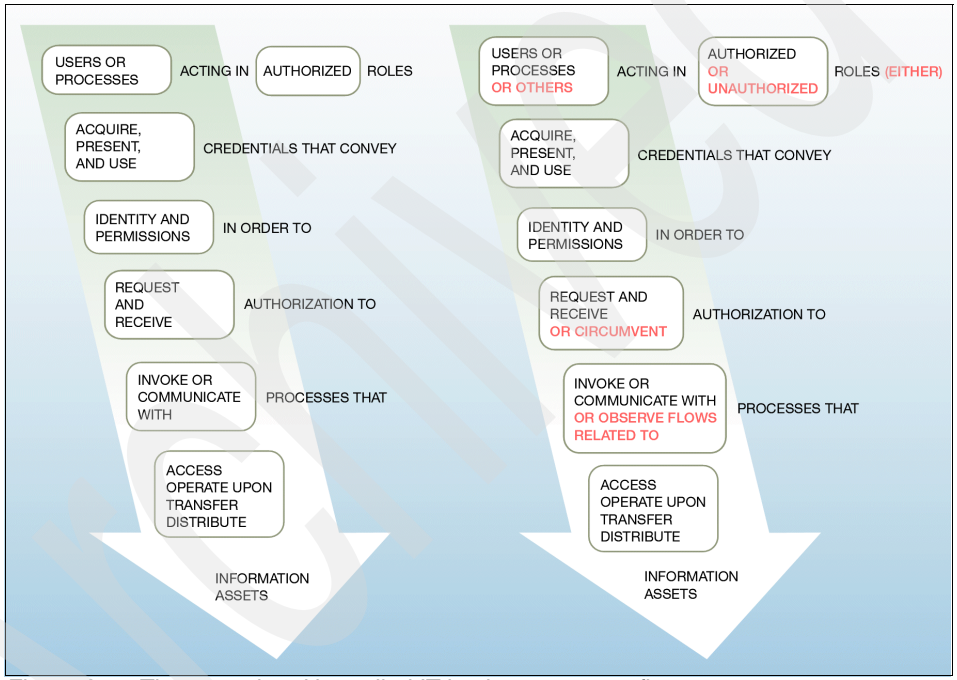


Figure A-8 The normal and imperiled IT business process flow

Security design objectives

Traditionally, security requirements have been expressed by referencing the security services within the OSI model: authentication, access control, data confidentiality, data integrity, and non-repudiation. This practice introduces ambiguity when applied in the context of business processes. This ambiguity can contribute to a miscommunication of security requirements and a mismatch of functionality within the computing solution. As with other architecture disciplines, the technical objectives of the security design activity need to be articulated in

quantifiable terms. Specific design objectives need to be developed and validated for each solution. For reference in this project, the following set of security design objectives were derived as a result of an analysis of the security-incident handling and reporting system for one corporation:

1. There is a need to control access to computer systems and their processes, consistent with defined roles and responsibilities.
2. There is a need to control access to information, consistent with information classification and privacy policies.
3. There is a need to control the flow of information, consistent with information classification and privacy policies.
4. There is a need to manage the reliability and integrity of components.
5. There is a need for protection from malicious attack.
6. There is a need for trusted identity to address the requirement of accountability of access to systems, processes, and information.
7. There is a need to prevent fraud within business processes and transactions, or to detect and respond to attempted fraud.

Selection and enumeration of subsystems

The security design objectives and the solution environment have a central role in the selection and enumeration of subsystems. Table A-2 shows a possible mapping of the example design objectives to security subsystems. It indicates where a subsystem may be required (R) or supplementary (S) in satisfying an individual security requirement. Actual subsystem selection requires documented rationale.

Table A-2 Mapping design objectives to security subsystems

Security design objectives	Audit	Integrity	Access control	Flow control	Credentials / Identity
Control access to systems/processes	S	S	R	S	S
Control access to information	S	S	S	R	R
Control the flow of information	S	S	S	R	S
Correct and reliable component operation	S	R	S	S	S
Prevent/mitigate attacks	R	R	R	R	S
Accountability through trusted identity	R	R	S	S	R
Prevent/mitigate fraud	R	R	R	R	R

There are many interrelated factors that determine how many instances of a given subsystem appear in the solution. Table A-3 suggests motivations for instantiating security subsystems within a design. Actual subsystem enumeration requires documented rationale.

Table A-3 Determining the security subsystems in a design

Subsystem	Number in a design	Characteristics of the computing environment
Security audit subsystem	Few	One subsystem for archive of related critical data One subsystem for analysis of related anomalies One subsystem for fraud detection in the solution
Solution integrity	Few	One subsystem per group of related critical components
Access control	1 to n	One subsystem per unique user-subject binding mechanism or policy rule set
Flow control	1 to m	One subsystem per unique flow control policy rule set One or more flow control functions per OSI layer service: physical, datalink, network, end-to-end transport, and application One or more flow control functions per domain boundary
Identity and credentials	1 to k	Some number of credential systems per domain Some number of disparate credentials or uses for credentials per domain Some number of aliases/pseudonyms at domain boundaries

Documenting conceptual security architecture

Given the agreed-upon design objectives, a conceptual model for security within the IT solution can be created. Figure A-9 and Figure A-10 on page 970 represent a conceptual security architecture. For clarity, security functions have been grouped by design objective.

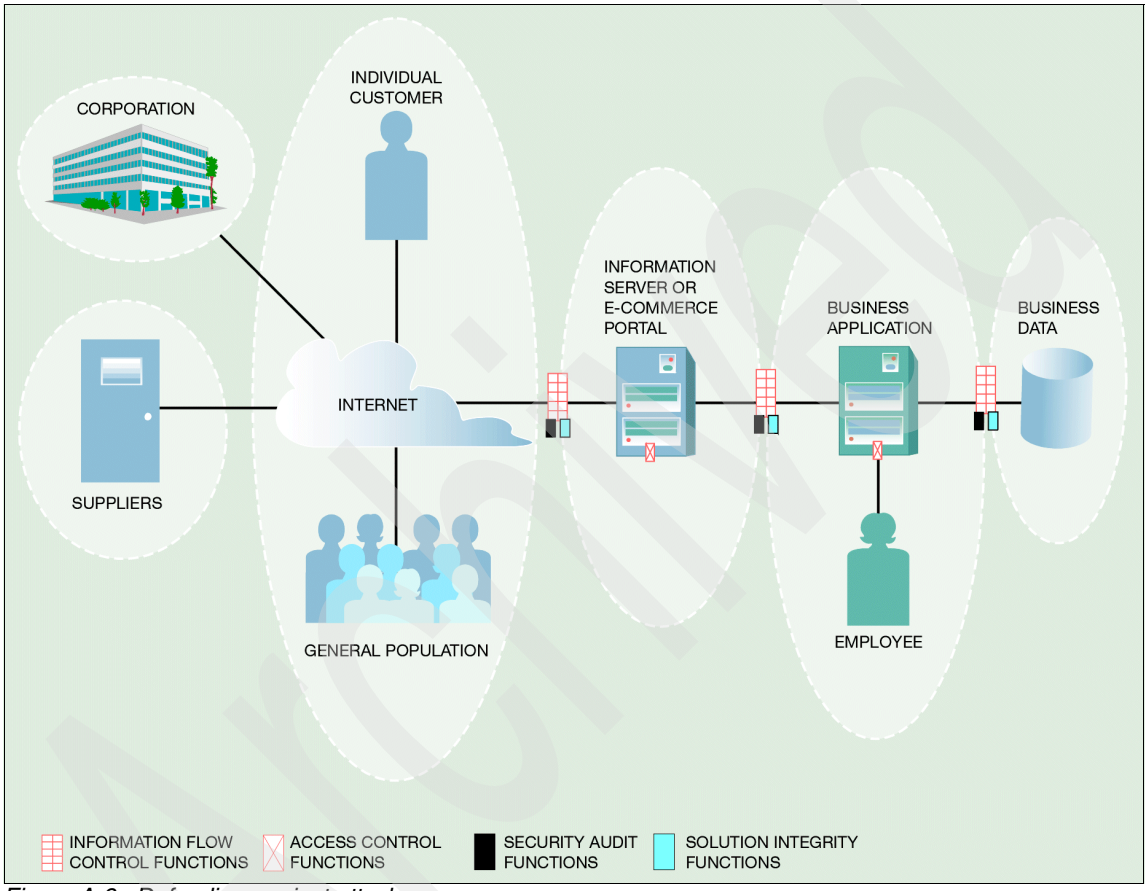


Figure A-9 Defending against attacks

The diagrams represent the solution environment segmented by risk profile or operational affinity, along with icons for security functions. The legend for the diagrams maps the security subsystems to icons. The information flow control subsystem has a wide range of functions. For this reason, a rectangle is used to indicate a policy evaluation and enforcement function, while an oval indicates a data flow function, such as a communication protocol with security capabilities.

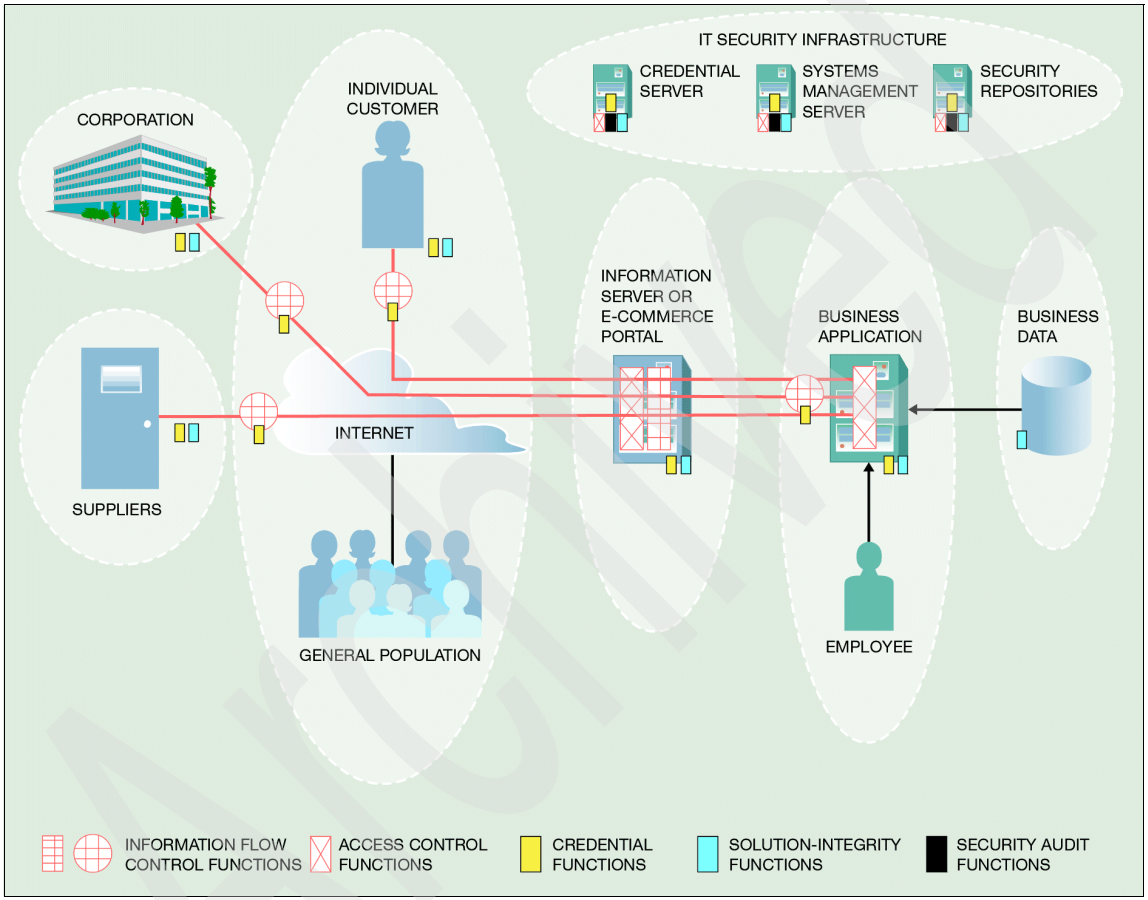


Figure A-10 Ensuring correct and reliable operation

From the perspective of the enterprise deploying the solution, the security design objectives will dictate where security functionality is desired; however, the compliance to some or all of the security requirements may be limited by the enforceability of policies beyond the boundaries of the enterprise. Whether and how these credential subsystems and access control subsystems can be integrated into the security architecture can have a major impact on the trustworthiness of the solution as a whole. These issues and dependencies should be considered and documented within architectural decisions.

This type of conceptual model forms the baseline for developing and evaluating a proof-of-concept and further refinement of the functional aspects of security within the target environment.

Integration into the overall solution architecture

There are several steps involved in translating the conceptual security subsystem functions into component-level specifications and integration guidance. These include creating models of the solution environment, documenting architectural decisions, developing use cases, refining the functional design, and integrating security requirements into component architectures.

Solution models

Creating an initial solution model is a critical step in the design process. With skill and experience, one-of-a-kind solution models can be developed to fit a given set of requirements. For complex solutions, the practice of using templates derived from prior solutions is becoming commonplace.

The Enterprise Solutions Structure (ESS) provides a range of reference architectures[4] for e-business solutions.

Documenting architectural decisions

Previously, the notion of the duality of security design was described (that is, ensuring correct and reliable operation and protecting against error and maliciousness). Both motivations are based upon managing the business risks of the solution and of the environment. Risks represent the likelihood that an undesirable outcome will be realized from a malicious attack, unexpected event, operational error, and so on. Risks are either accepted as a cost of operation, transferred to some other party, covered by liability insurance, or mitigated by the security architecture.

Architectural decisions will dictate how robust the security system architecture should be, which security subsystems to incorporate into the system architecture, which functions and mechanisms within each subsystem should be deployed, where the mechanisms will be deployed, and how the deployment will be managed.

[4] P. T. L. Lloyd and G. M. Galambos, “Technical Reference Architectures,” IBM Systems Journal 38, No. 1, 51–75 (1999).

Examples of architectural decisions include the following:

- Viability of the countermeasures, including the threats addressed, the limitations and caveats of the solution, and the resulting window of risk
- Extensibility of the design, including whether the design will serve the total population and whether there will be separate designs for defined population segments
- Usability of the design, including whether the mechanisms integrate with the technology base and the extent of the burden of compliance for users
- Manageability of the design, including the extent of the burden of lifecycle management

Use cases

Architectural decisions will also drive the evaluation of prototypes and models of functions within the solution. One form of prototype is called a *use case*. Both security threats and normal interactions and flows can be validated with use cases.

Example 1: Interception of errant packet or message flow

Figure A-11 on page 973 represents several levels of detail for the operation of an information flow control subsystem that is designed to monitor, send, and receive operations that cross a boundary between two networks.

The computer systems are represented in the physical view. In the component view, an information flow control interface, positioned between source and destination, will examine one or more aspects of packets or messages sent across the boundary. Some components of this information flow control subsystem are shown in the logic view, where the monitored conditions and the programmed actions are carried out, based on a set of rules.

Valid packets are allowed to flow across the boundary; however, packets or messages of a specified format, or from an invalid source, or to an invalid destination, are disabled by the security subsystem. A record of the event is generated by invoking an interface to a security audit subsystem.

This example is representative of the type of filtering, analysis, and response that is performed within packet filter firewalls or electronic mail gateways.

There are many architectural decisions to be evaluated within each iteration of the design. The effect on performance due to processing delays, plus the effect of data collection and analysis on the overall operation of the solution, are significant factors.

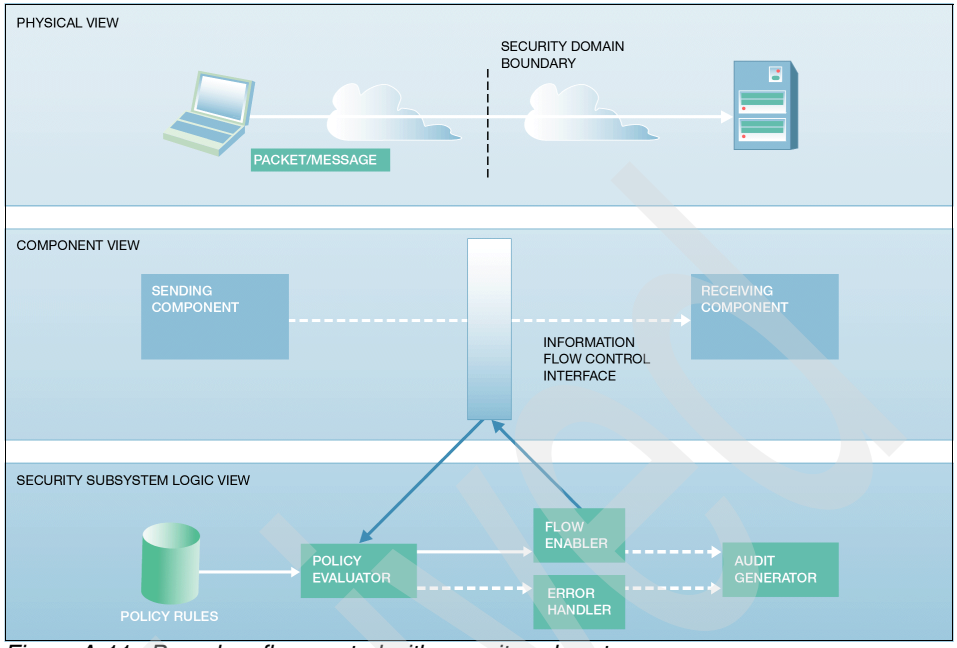


Figure A-11 Boundary flow control with security subsystems

Example 2: Three-tier client/server input flow

Figure A-12 on page 974 illustrates an input flow for a three-tier client/server process that is typical of the integration of enterprise computing with the Internet environment.

Several instances of security subsystems are depicted, spread among three network security domains. An information flow control subsystem is positioned at the boundary points between networks. An access control subsystem is positioned between a receiving component and its corresponding application component. Interfaces to related credential subsystems and security audit subsystems are shown in the security subsystem logic view. No integrity subsystem functions are referenced in this example. The scenario follows:

1. The business process interface is invoked by a user or a process and the request is transferred via a sending component.
2. The request flows across a security domain in a manner that is acceptable to the sending and receiving components, based on the defined information flow control rules.
3. Identification, authentication, and access control decisions are made based on the external identity associated with the request by an access control subsystem associated with the middle-tier application.

4. The middle-tier application is invoked via a user-subject binding. The actual processing is not covered here; it may include business presentation and data mapping logic, or it may be performed by an application-level information flow control subsystem, such as a proxy server.
5. The middle-tier application initiates, or relays, a request to the end-tier application. The request is scrutinized at another network boundary control point.

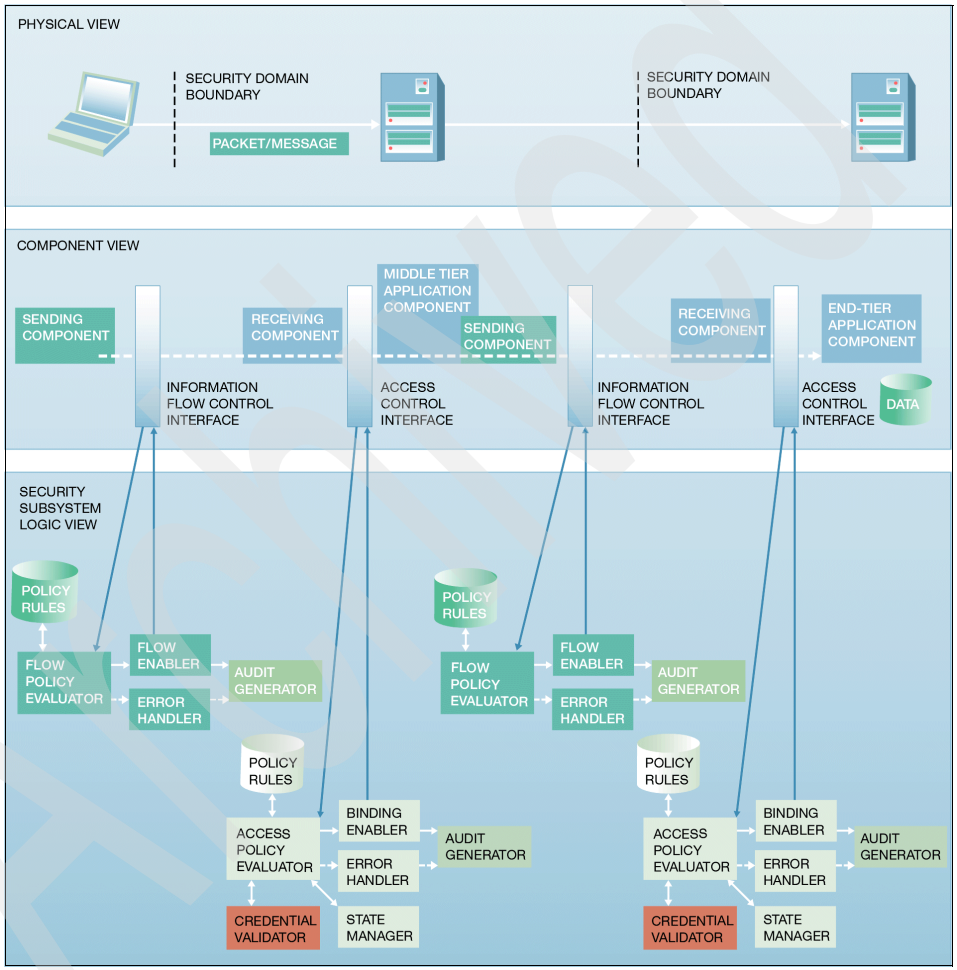


Figure A-12 Three-tier client/server input flow with security subsystems

6. At the end-tier application, an access control decision may be performed on the request relative to the identity of the user represented by the middle-tier application, depending on the design of the application and the exchange protocols used.

7. The business process is invoked by a user-subject binding if the access control decision is positive.

This demonstrates how security functions from several subsystems are distributed throughout the solution. As with the first example, architectural decisions will guide the design of the security subsystem functions, which in turn may put constraints on the overall business flow in order to achieve the risk management objectives.

Refining the functional design

Walk-throughs of complete business processes, including exception conditions and handling processes, assist in creating a viable solution outline and refining requirements and interdependencies among the solution building blocks.

Example 3: PKI digital certificate enrollment

This example uses the credential subsystem model to describe the generalized flow for enrolling a user into an identity or credential system based on PKI digital certificates as the first step in developing a security system architecture. The process involves combining the subsystem model with assumptions about the business environment, the business processes, the risk management requirements, the technical specifications, and possibly the legal and business compliance requirements associated with issuing PKI digital certificates.

In Figure A-13 on page 976, the yellow blocks represent manual processes, the blue blocks map to automated processes, and the peach blocks map to automated audit data capture points. The blue data storage icons represent sensitive repositories, the pink icons map to cryptographic secrets, the white icons represent unique contents of the certificate, and the lavender icon is associated with the certificate.

The enrollment process flow depicted demonstrates the exchange of sensitive user information and secrets, plus the export of the credential outside the control of the issuer. The full enrollment scenario should include processes from a corresponding information flow control subsystem. For public key credentials, the format of certificates, along with details of how the credentials are formatted, transported, and stored, are important design considerations. All scenarios must be validated against existing and proposed business processes. Validation of the scenarios substantiates the architectural decisions discussed earlier. Subsequent design steps are needed to develop and map the functions of the security subsystems to Common Criteria specifications and ultimately onto the nodes and physical components.

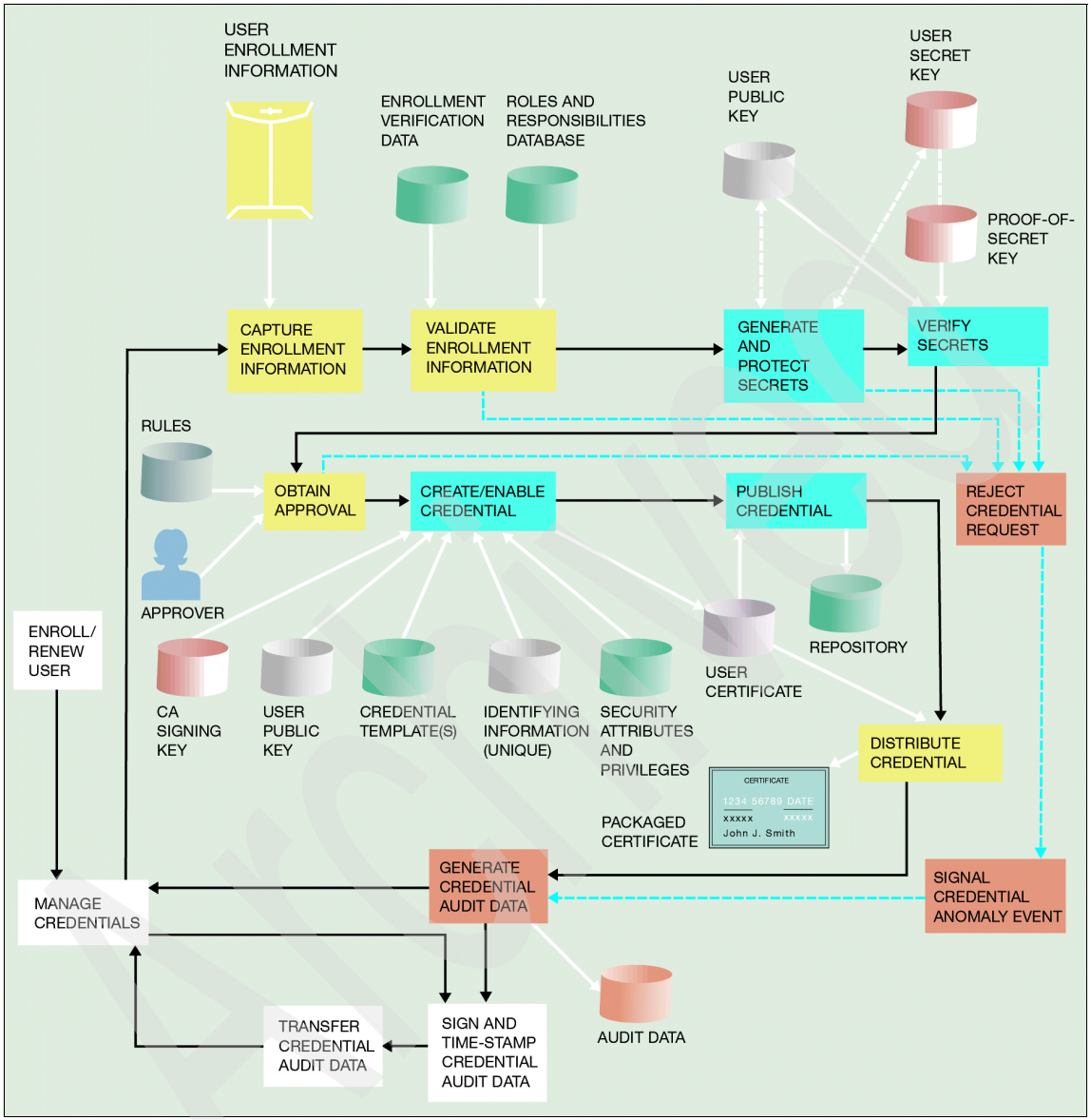


Figure A-13 Sample PKI digital certificate enrollment process flow

Integrating requirements into component architectures

The security functions within the design need to be apportioned throughout the solution. However, many of the mechanisms and services within the IT solution that implement security functionality operate within other than security

components, for example, database systems, application systems, clients, servers, and operating systems. The task of adopting security functions into the network, application, middleware, security, systems management, and infrastructure architectures is shared by the several architects and integration specialists involved in the design project. The process involves a structured approach, considering the purposeful allocation of functions and requirements throughout the component architectures by:

- Mandate, based on a legal or contractual compliance requirement
- Best practice for security, or for balance of security and business process
- Component capability, knowing the existence of a mechanism that supports the required process or action
- Location in the configuration, based upon interaction with components or flows
- Impact, considering the risk, security objective, or the component capacity to perform
- Necessity, because there may be no better alternative

Summary of the design process

This section has described the process for translating the conceptualized security solution into a set of detailed specifications, for an integrated IT security control system, using the security subsystem construct. The design is documented, refined, and validated against the business processes through use cases and scenarios. The detailed security requirements, expressed in terms of Common Criteria component-level detail, are distributed throughout the operational model for the IT solution. At this point, integration-level detail can be finalized, and the implementation plan can proceed.

Conclusions

This appendix has examined the issues and circumstances that affect the design of comprehensive security functions for computing solutions. It has outlined a system model and a systematic process for security design with the Common Criteria international standard at its foundation.

Several summary observations can be made relative to this proposed model and process:

- Security is a shared responsibility among all IT design disciplines.
- Security design is linked to business objectives beyond the need for protecting against attack, and conversely, protecting against attack does not in itself meet all the business requirements for security.
- Many, if not most, security control points within IT solutions are found in portions of solutions that are not typically considered security components.

Reliable and correct operation of solutions using secure data exchange protocols, such as IPSec and SSL, is predicated on functions within all five of the security subsystems defined in the proposed model and design process. These protocols are based on trusted identities that utilize cryptographic keys requiring storage integrity, reliable key exchange protocols, strong access control mechanisms, reliable data exchange protocols, and trusted audit trails for enrollment and key lifecycle management. Furthermore, the proposed model provides a new perspective for viewing Common Criteria protection profiles in the context of security subsystems. For example, the protection profile for an application gateway firewall suggests the functionality of all five security subsystems. The fact that a front-line security device, such as a firewall, might fit the definition of a credential subsystem highlights the critical nature of its design, integration, and operation.

Actions and further study

The concepts and the supporting detailed information presented in this appendix were incorporated into training for IBM Global Services architects. Additional work is underway to develop notations, models, and visualization techniques that enhance its adoption in related methods and architect disciplines. A patent application has been filed for the system and process, designated Method for Architecting Secure Solutions, or MASS.

Several of the notations, models, and visualization techniques will be applied throughout this book.

Global MASS: An example

In our final section of this appendix, we present a practical example of applying the Method for Architecting Secure Solutions (MASS) within an e-business IT infrastructure.

As described in this appendix, MASS works with standard components to represent the security solution in a formal way. MASS starts from a high-level view, based on the business requirements, and ends with a detailed (but still non-product or technical-related) view. This chapter discuses the four steps of this process.

Business view

In this view, we focus on the elements that are defined by the company's business needs: the customers, as well as the internal staff, have to work on the ordering system. The customer is located in an area that is totally *uncontrolled* by the company, and the internal employees are working on the company intranet. This section of the network is controlled and its access is *restricted*. The ordering system is located in the most *secured* part of the network.

Despite several levels of *access control,* a flow must be established and controlled among the three components that belong to the same *community* performing e-business transactions.

The view depicted in Figure A-14 on page 980 integrates all the aspects of the solution we have described. In the next step, we list the components that are needed to achieve the security solution in a more detailed way.

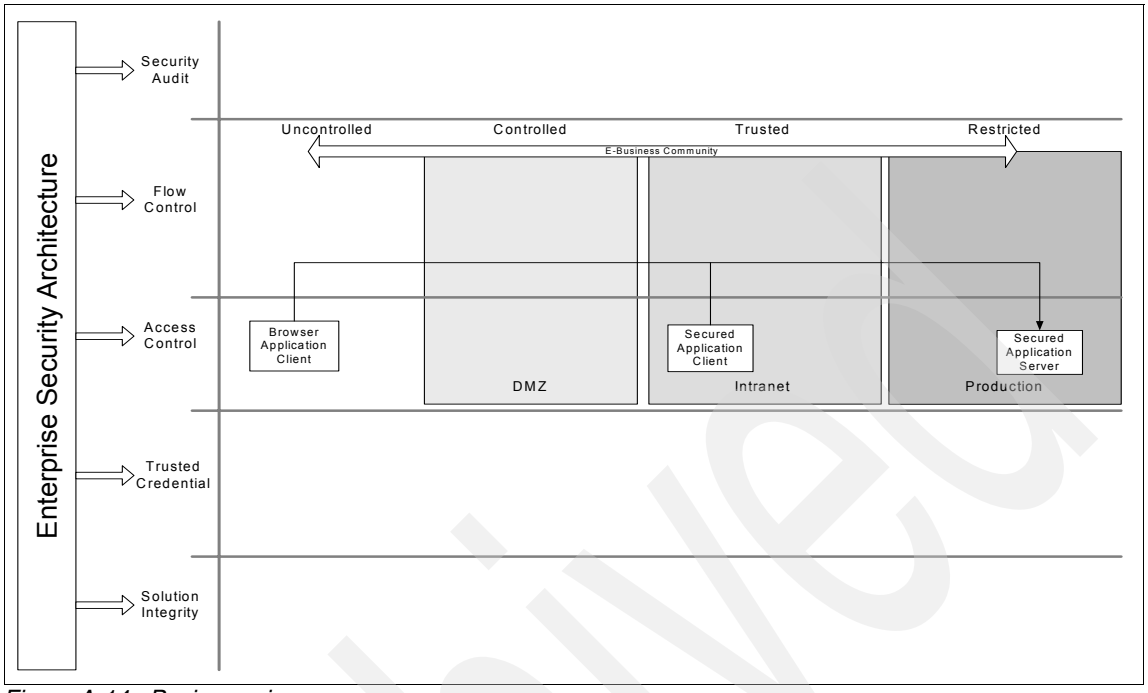


Figure A-14 Business view

Logical view

The logical view is now applying some security concepts to the previous one, such as:

- Different level of network security
- Portals
- SSO

Two new areas will also be introduced:

- Auditing
- Integrity

Although the customers, the employees, and the ordering system are part of the same community, they are obviously not connected directly. They belong to areas that offer different levels of trust. The e-business community is composed of subcommunities with different levels of trust and the equivalent security measures.

The *external community* is the least trusted and controlled one, and the systems that are located in this community are uncontrolled. In order for these systems to

become part of the overall e-business community, they have to verify their identities by using specific authentication mechanisms, that is, user authentication within the Web application.

The *managed communities* are considered controlled because there are at least basic control mechanisms in place to monitor access in this area. Any system located in this community can be considered an “authorized system” located on the intranet.

The *closed community* contains critical systems where a high level of control is applied. Dynamic Host Configuration Protocol (DHCP) is, for example, not allowed in this community.

The customers are part of the external community. Because this system is not considered trusted, the communication channel must be secured and the user must be authenticated. This is done within the controlled area, which is a demilitarized zone (DMZ) between the uncontrolled area and the restricted area. A new component has to be added in order to deliver the necessary functionality: a Web portal working as an interface between the remote system and the internal secured ordering system.

It is obvious that, due to the nature of the information exchanged, the data integrity must be preserved. Also, auditing functions are needed to supervise the external and internal communication in order to log all transactions within the ordering system. A single sign-on system is needed to propagate user authentication and credential information from one system to the other.

The different components are shown in Figure A-15.

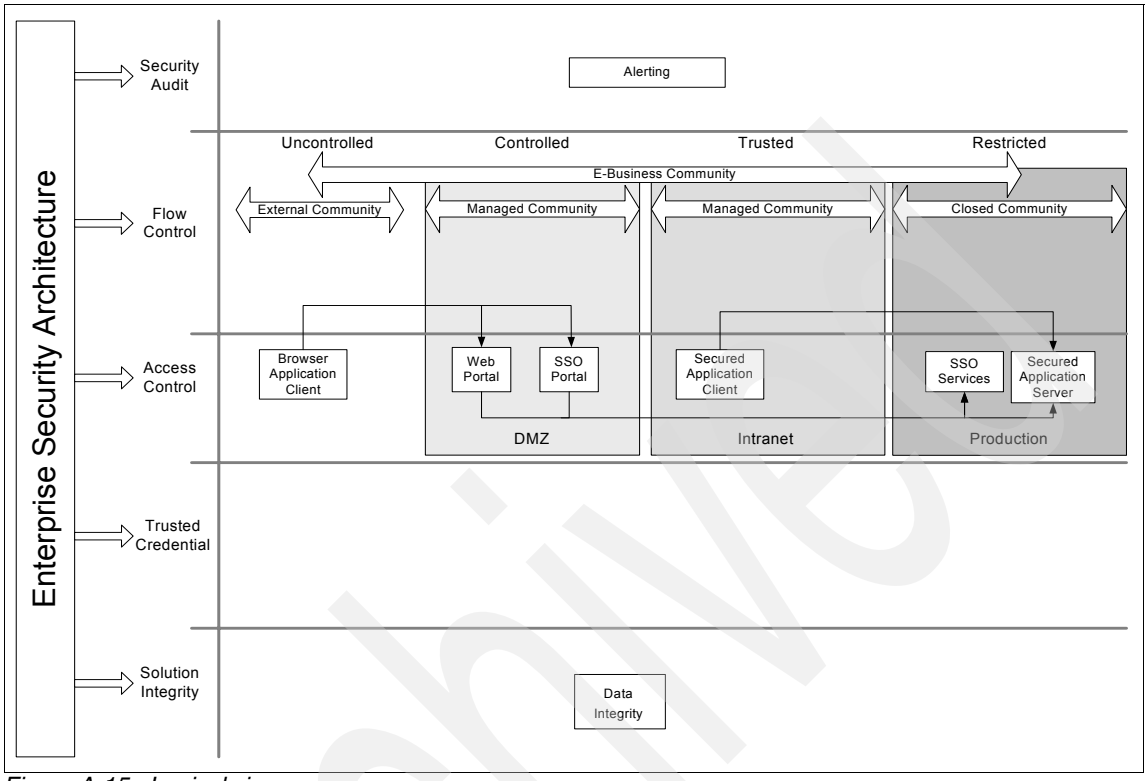


Figure A-15 Logical view

Detailed view

We have introduced the basics of the functions that we are using, such as SSO, Web portal, data integrity, and so on. This section describes them in more detail, and Figure A-16 on page 983 illustrates the picture.

The access and flow control sections now deal with the need to interface the various areas with their different levels of trust. Therefore, *boundary components* are added. These components can be implemented as a firewall or network segregation function. Another function of a boundary component is the SSL terminator that is located between the uncontrolled and the controlled community.

The *security audit* subsystem collects two specific types of elements: *event logging* receives life events from active devices and applications, and *component logging* gets information from sources such as other log files or active network

scanners, and so on. This results in the delivery of reports and alerting or reactions.

Another level of detail is added in terms of attachment type. This describes the actual type of connectivity, mainly whether it is static or managed (for example, dynamic IP addressing, when using DHCP services for intranet systems).

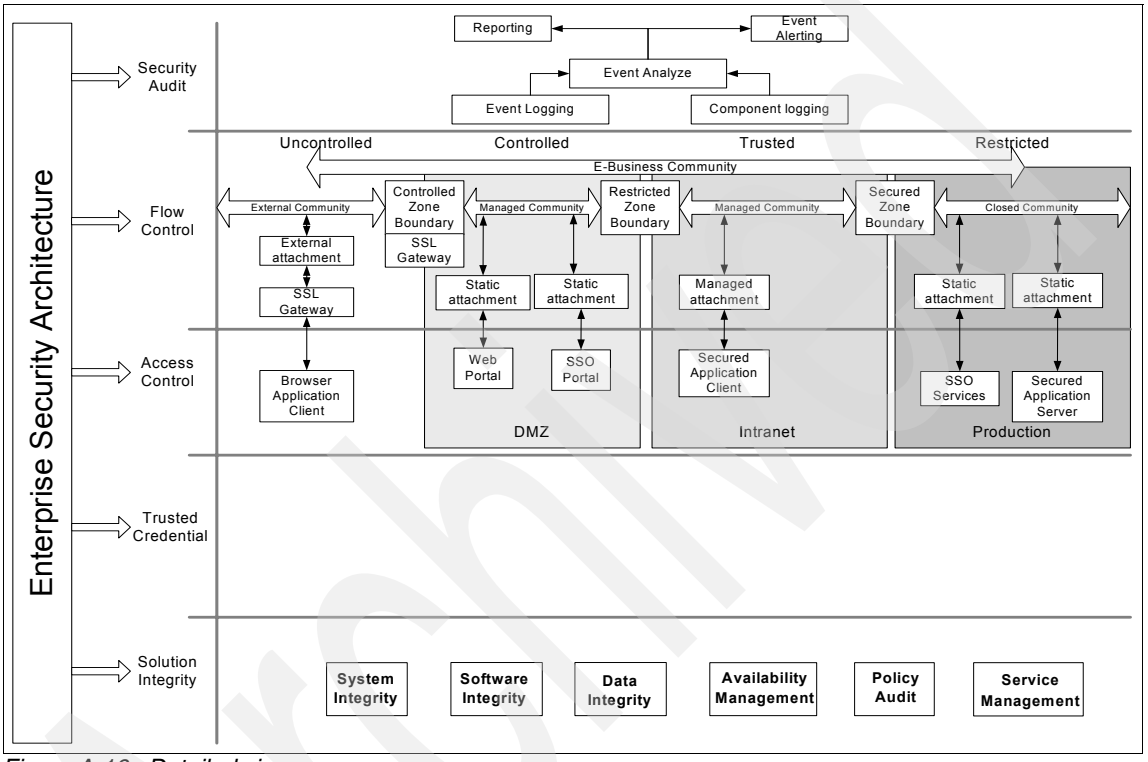


Figure A-16 Detailed view

In terms of solution integrity, these are all of the components. The basic requirements are system and software integrity for the systems, data integrity for the transactions, and the management of the availability of the system, as well as other services required by the e-business application. Finally, the policy audit must be in place to comply with the overall enterprise security policy.

Full architectural view

We have added all of the components except in the *Trusted Credential* subsystem. This section represents the workflow and the components applied to the credential management. It describes the process flow when a user tries to obtain new access authorization: the request, the validation of the request

against the rules, and the creation of the credential and its distribution to the authentication systems and end-points and their storage. The credential subsystem also relies on the corporate user management system represented as a related component.

This architectural approach is representing a very simple example in which only a few of the components were actually used. However, it already shows that it globally addresses the security design objectives of an enterprise e-business architecture. It shows the flow and access controls systems and brings into perspective the auditing function that is mandatory in today's security infrastructures. Adding the credential and the integrity subsystem into one global picture enables the architect to depict all of the necessary components relevant to an overall enterprise security architecture, as shown in Figure A-17.

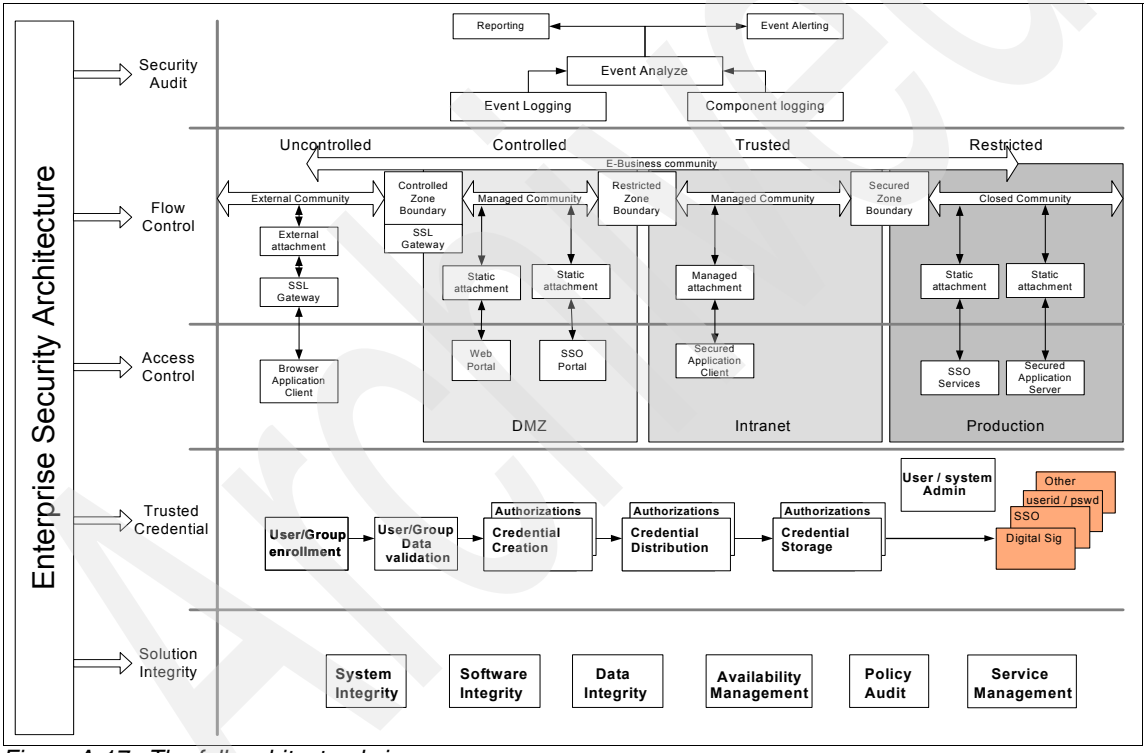


Figure A-17 The full architectural view

B

Productivity and functional enhancements

In this appendix we discuss several productivity and functional enhancements for some of the Tivoli security products that are publicly available but are not supported by IBM.

What does this mean?

These tools were developed by IBM Tivoli or other companies and business partners and are being made publicly available[1] for general use but have not been included into the extensive product test phases that are necessary in order to officially support a new version or release of a software.

We cover the following tools and enhancements in this appendix:

- Tivoli Identity Manager Adapter Development Tool
- Tivoli Identity Manager Graphical Configuration Editor
- Tivoli Identity Manager Monitoring Solution
- Documentation Tool for Tivoli Identity Manager

[1] These tools can be obtained from the IBM Tivoli Open Process Automation Library. Each downloadable package has an applicable user license agreement. The terms and conditions under which a specific item may be used and the responsibilities of the user and the item provider are described in the end user license agreement for that item. You can find the IBM Tivoli Open Process Automation Library at http://catalog.lotus.com/wps/portal/topal

- Tivoli Identity Manager Data Feed Reports
- Tivoli Access Manager Monitoring Solution

Tivoli Identity Manager Adapter Development Tool

The Adapter Development Tool, ADT, is a tool used by Tivoli Identity Manager clients and consultants to create custom Identity Manager adapters. It reduces adapter delivery time and helps in the development of custom adapters by doing the following:

- Providing graphical development that integrates Directory Integrator functionality and Identity Manager profile development
- Reducing errors caused by manual editing of files
- Providing automated validation to identify common errors
- Providing templates of adapter customizations
- Allowing export and import of adapters in either DSML or RMI formats
- ADT produces Directory Integrator RMI adapters—the defacto standard

The Identity Manager Adapter Development Tool (ADT) is built using the Eclipse 3.1 platform and is a Rich Client Program (RCP). All required Eclipse components are packaged with the application. Eclipse does not need to be installed on the target platform.

A Java runtime version 1.4 or greater is required for installation and operation.

This application makes use of the Tivoli Directory Integrator version 6.0 product. Directory Integrator must be installed on the platform prior to installing this tool and the location of the Directory Integrator home must be provided during the installation process.

Tivoli Identity Manager Graphical Configuration Editor

With Graphical Configuration Editor (GCE), a Tivoli Identity Manager configuration can be captured from an active Identity Manager installation and transferred to an off line environment. The Identity Manager configuration can then be edited in this environment. If desired, the edits can then be pushed back to the Identity Manager installation.

GCE provides additional views of how the Identity Manager configuration's data elements interact. These views are unavailable in the standard Identity Manager user interface. GCE also provides validation for JavaScript and LDAP filters data

elements. Identity Manager lacks a validation facility for these data elements. GCE also allows the creation of templates, which represent common configurations to be captured and reused. Templates allows the client to make rapid and consistent changes to their environment.

GCE allows the client to save an Identity Manager configuration as a file. This file can be sent to IBM support for further analysis of a client's problems. This file could also be used as a baseline to configure upstream systems, such as a QA Identity Manager environment using the Identity Manager development environment as a starting point. GCE allows for a complete export of an Identity Manager configuration, unlike the limited import and export facility provided with Identity Manager. GCE is built on top of the Eclipse framework, which provides a familiar environment to a user.

Tivoli Identity Manager Monitoring Solution

The solution provides the capability of monitoring a Tivoli Identity Manager server using the Tivoli Monitoring 6.1 Universal Agent. The Tivoli Identity Manager Monitoring Solution uses the *File, Script, and Socket Data Providers* to extract useful metrics about the health of your Tivoli Identity Manager server.

A separate guide is included that provides detailed information about the metrics that the Agent collects and describes how to customize workspaces and scenarios that make use of the metrics.

This solution provides you with useful data about the performance characteristics of your Tivoli Identity Manager server including the following:

- Server availability and server process activity
- Memory usage characteristics: heap size before and after garbage collection, max heap size, garbage collection time
- Workflow queue backlog
- User page response times
- Tablespace usage
- Logged error messages

The collected information can be used to perform trending analysis on your Tivoli Identity Manager server. The Universal Agent runs on all platforms supported by the Universal Agent (Windows, AIX, Solaris, Linux and HP/UX systems) that can run shell, awk, and perl scripts.

Documentation Tool for Tivoli Identity Manager

DocTool runs on any Windows or Unix system that have access to the Tivoli Identity Manager LDAP directory, such as Tivoli Directory Server. The features and functions of DocTool are as follows:

- Documents Identity Manager configuration
  - Roles
  - Policies
  - Forms
  - And all other configuration elements
- Uses Identity Manager directory as input
- Single file output (HTML)
- Produces current state of Identity Manager configuration
- Level 1 and 2 support use to help clients
- Consultants quickly get full configuration documentation
  - First step in understanding an Identity Manager configuration and diagnosing anomalies
  - First step when entering an existing implementation
  - Last step when closing an engagement
- Clients understand their Identity Manager as implemented
- Easy to install and use

Tivoli Identity Manager Data Feed Reports

Tivoli Identity Manager identity feeds can result in the creation of hundreds or thousands of Identity Manager requests. When the identity feed is initiated from Tivoli Directory Integrator the Identity Manager requests run asynchronously. The results of the requests are not communicated back to Directory Integrator, so there is no easy way for the feed's AssemblyLine in Directory Integrator to report on the success of the feed.

Identity Manager provides reporting capabilities that can be used to show the states of any *person add*, *modify*, and *delete* requests that were created by an identity feed. But running such a report requires manual steps on the part of a user. And although the report can tell you that a person request failed or returned a warning, it cannot drill down into the request's sub-processes to find the root cause of the problem.

The result is that identity feeds are often run with no monitoring of their results. It is assumed that the feed is working correctly until someone complains that an identity is missing or incorrect. When it is found that investigation of the feed is necessary, it is often accomplished by tedious scanning of Identity Manager's completed requests logs.

These Directory Integrator assembly line scripts can automatically generate reports showing the results of Identity Manager identity feeds. These scripts can be combined with existing identity feed configurations with minimal changes to the existing feed configuration. Reports can be created in either XML or HTML formats.

Tivoli Access Manager Monitoring Solution

The solution provides the capability of monitoring a Tivoli Access Manager server using the Tivoli Monitoring 6.1 Universal Agent. The Tivoli Access Manager Monitoring Solution uses the HTTP, File, and Script Data Providers to extract useful metrics about the health of your Tivoli Access Manager server.

A separate guide is included that provides detailed information about the metrics that the Agent collects and describes how to customize workspaces and scenarios that make use of the metrics.

This provides you with useful data about the performance characteristics of your Tivoli Access Manager server including the following:

- Server availability and server process activity
- WebSEAL statistics
- Junction statistics
- Response times
- Workload

The collected information can be used to perform trending analysis on your Tivoli Access Manager server. The Universal Agent runs on all platforms supported by the Universal Agent (Windows, AIX, Solaris, Linux and HP/UX systems) that can run shell, awk, and perl scripts.

Conclusion

In this appendix, we introduced some tools that enhance functionality of Tivoli Identity Manager and Tivoli Access Manager deployments. They can save you

valuable time during deployment and help you to better manage an Identity Manager or Access Manager environment, independently of its size.

As Identity Manager and Access Manager grow in popularity, more IBM Business Partners are offering solutions that will complement them, and this is especially true when you see Identity Manager and Access Manager as the framework for your Identity and Access Management security environment.

Glossary

802.11a, 802.11b, 802.11g Three IEEE substandards for wireless local area network (LAN) technologies. These provide for varying transmission speeds of 11 to 54 Mbps (which in reality translate to raw throughputs of roughly 6 to 30 Mbps).

access control list (ACL) A cornerstone of security is the ability to determine who can access computer networks and systems. Control can be exercised through the use of access control protocols, computer applications that authenticate the user logging into a network. ACLs define which users can access specific data and programs. Access codes are passwords, series of characters or numbers that enable a user to access the network.

ACL See access control list.

Active Requestors An application (possibly a Web browser) that is capable of issuing Web services messages such as those described in WS-Security and WS-Trust.

ActiveX® The name Microsoft has given to a set of "strategic" object-oriented programming technologies and tools. The main technology is the Component Object Model (COM). Used in a network with a directory and additional support, COM becomes the Distributed Component Object Model (DCOM). The main thing that you create when writing a program to run in the ActiveX environment is a component, which is a self-sufficient program that can be run anywhere in your ActiveX network (currently a network consisting of Windows and Macintosh systems). This component is known as an ActiveX control. ActiveX is Microsoft's answer to the Java technology from Sun Microsystems. An ActiveX control is roughly equivalent to a Java applet.

AEF Access Enforcement Point.

agent A function that represents a requester to a server. An agent can be present in both a source and a target system.

API See application programming interface.

application programming interface (API) Software applications, such as spreadsheets or word processing, use a special language and message format, the API, to communicate with the computer operating system, database management system, or other system programs.

assertion In computer programming, a programming language construct which immediately aborts program execution if a certain condition or expression is false (an "assertion failure"). It is used by programmers during development to check for potential errors or bugs. To assist with this, the implementation of assertions in many languages will provide information such as the filename and line number in the source code that triggered the assertion failure.

association The process by which principals become associated or affiliated with a trust realm or federations.

assurance The determination that host platforms, end-user platforms, applications, network component configurations, and operations are in accordance with security policy. Entities are monitored to ensure policies have been implemented and used. Detected noncompliance with policies is recorded and reported. Remediation of policy noncompliance is based on remediation policy.

asymmetric keys In computer security, the two keys in a key pair. The keys are called asymmetric because one key holds more of the encryption pattern than the other does.

attribute service A Web service that maintains information (attributes) about principals within a trust realm or federation. The term principal, in this context, can be applied to any system entity, not just a person.

audit The recording of security events in a log. To ensure future claims that security events recorded are accurate and have non-been altered (that is, non-reputable), audit records are collected and secured. Audit records may be used for:

- Internal problem analysis.
- Use as evidence in relation to a potential breach of contract, breach of regulatory requirement or in the event of civil or criminal proceedings, for example, under computer misuse or data protection legislation.
- Negotiating for compensation from software and service suppliers.

Audit logs are created by system components, including operating systems, applications, and network devices.

authentication Authentication denotes a security procedure where an individual is identified. The process ensures that the individual is whom he or she claims to be, but does not affect the individual's access rights. User names, passwords, and biometric scanning are all authentication techniques.

authorization This phase of security admits only legitimate user access to systems, data, applications or networks. After the user is authenticated, he is authorized, that is, granted access to a network resource. An identification number or password that is used to gain access to a local or remote computer system.

anti-virus management Anti-virus (AV) clients run on host platforms. Anti-virus management includes:

- AV client distribution and updates to authorized platforms. Platforms may be initially loaded with AV clients or fetch AV clients from the AV manager.
- Notification that updates are available.
- Making AV clients and updates available for automatic download when the host platform connects to the manager.
- Receiving host AV log files and host AV configuration data.
- Providing summary AV event information and alerts.
- Providing reports.

B2B Business to business.

B2C Business to consumer.

B2E Business to employee.

Basel II Central bank governors and the heads of bank supervisory authorities in the Group of Ten (G10) countries issued a press release and endorsed the publication of *International Convergence of Capital Measurement and Capital Standards: a Revised Framework*, the new capital adequacy framework commonly known as Basel II. More information can be found at http://www.bis.org/publ/bcbsca.htm

Binding Security and Secure Conversation Security binding is the protocol that ties security attributes together, such as an identity and the authorizations for the identity. Examples of security bindings are:

- Secure Sockets Layer and Transport Layer Security protocols provide for the secure authentication of servers and clients.
- X.509 certificates bind an identity to a public key.
- A Web cookie binds an identity to a service.

Security conversion securely maps information from one form to another form. For example, a password and ID may be converted to a common format for an authenticated identity. Confidentiality may convert plaintext information into cipher text using an encryption key or keys.

BS7799 British Standard 7799, a document describing enterprise security.

CA See Certificate Authority.

CDC Common Domain Cookies.

certificate The most common kind of credential in the network computing environment. Certificates include standard information such as the owner's public key, globally accessible name, and expiration dates; certificates may also contain some application-unique data such as title, degree(s) earned, and professional licenses. Certificates are also called digital certificates.

Certificate Authority (CA) In the pre-Internet world, every secure transaction involved a trusted third party—such as a notary, attorney or broker—who could guarantee that both parties were who they purported to be. A Certificate Authority fills that same role in the digital world.

An authority in a network issues and manages security credentials and public keys for message encryption. As part of a public key infrastructure (PKI), a CA checks with a Registration Authority (RA) to verify information provided by the requestor of a digital certificate. If the RA verifies the requestor's information, the CA can then issue a certificate.

A CA vendor, such as VeriSign or Entrust, issues certificates that contain the identities and affiliations of individuals, along with their public keys. These certificates are bound together with the digital signature and stored in a special directory. The sender's browser looks up the recipient's certificate in the directory, and the message can be encrypted using the key embedded in the certificate. The sender can then sign the message using his own private key, and the recipient can verify the signature by using the sender's public key that is vouched for by the CA.

CGI See Common Gateway Interface.

Circle of Trust The group of service providers that share linked identities and have business and operating agreements in place is known as a circle of trust.

claim A declaration made by an entity (for example, name, identity, key, group, privilege, capability, attribute, etc.).

claim confirmation The process of verifying that a claim applies to an entity.

Common Gateway Interface (CGI) A specification for transferring information between a World Wide Web server and a CGI program. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, Java, or Visual Basic. CGI programs are the most common way for Web servers to interact dynamically with users. Many HTML pages that contain forms, for example, use a CGI program to process the form's data once it is submitted.

Common Object Request Broker Architecture (CORBA) An architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker." CORBA was developed by a consortium of vendors through the Object Management Group, which currently includes over 500 member companies. Both the International Organization for Standardization (ISO) and X/Open have sanctioned CORBA as the standard architecture for distributed objects (which are also known as components). CORBA 3 is the latest level.

container A Java run-time environment for enterprise beans. A container, which runs on an Enterprise JavaBeans server, manages the life cycles of enterprise bean objects, coordinates distributed transactions, and implements object security.

CORBA See Common Object Request Broker Architecture.

credential exchange The purpose of a credential subsystem in an IT solution is to generate, distribute, and manage the data objects that convey identity and permissions across networks and among the platforms, the processes, and the security subsystems within a computing solution. Credentials are created as a result of a successful authentication. Some common types of credentials are:

- X.509 public key identity certificates that bind an identity to a public key.
- X.509 attribute certificates that bind an identity or a public key with some attribute.

Kerberos tickets that are encrypted messages binding the holder with some attribute or privilege, and encrypted cookies.

credentials Data associated with a user or resource that indicates identity and authority level. Credentials need to be issued by a trustworthy authority, as that authority is vouching for the identity and authorization level. A passport is a credential; it represents the bearer's identity and rights and is issued by a formally recognized government agency. In network computing environments, the most common type of credential is a certificate that has been created and "signed" by a trusted Certificate Authority.

CTCPEC Canadian Trusted Computer Product Evaluation Criteria published by the Canadian government.

CUID Common Unique Identifier.

Data Protection Act 1998 (U.K.) An Act to make new provision for the regulation of the processing of information relating to individuals, including the obtaining, holding, use or disclosure of such information. More information can be found at http://www.opsi.gov.uk/acts/acts1998/19980029.htm

demilitarized zone (DMZ) An area of your network that separates it from other areas of the network, including the Internet.

DHCP See Dynamic Host Configuration Protocol.

Digital Certificate Digital certificates allow a user to send an encrypted message. A digital certificate is an attachment to an electronic message that verifies the user is who one claims to be, and is used to ensure secure e-business transactions. The Certificate Authority (CA), which issues a user's digital certificate, makes known the user's public key, which another user employs to decode the digital certificate attached to a message. This process also verifies that the certificate was issued by the CA and allows users to obtain identification information of the certificate-holding sender. The recipient of the message can then send an encrypted reply.

directory A directory service is the "yellow pages" of computer network resources, stored on a server and often containing security-related data, such as phone numbers, e-mail addresses, public keys, computer names, and addresses. The data is presented hierarchically, much like a family tree, with one section providing key information about the files beneath it. To access a file, a user may need to produce the names of all the directories above it by specifying a path. To read information from or write information into a directory, the user must use operating system commands.

Directory Services Provide means of locating resources and users in a network or networks. They are analogous to telephone directories—even though you look up a resource or user name, you still need to know something about its location to narrow the search. A directory can also include the public key of the user or resource in addition to location and other information.

Directory Services Markup Language (DSML) An application of the Extensible Markup Language (XML) that enables different computer network directory formats to be expressed in a common format and shared by different directory systems.

In the latest DSML specification, the related XML schema defines types of information found in today's network and enterprise directories. It then defines a common XML document format that should be used to display the contents of each directory.

DSML has been heralded in industry press as a key component to the future of e-commerce and Web-based applications that link businesses and business processes together. Some examples of such business-to-business and business-to-customer applications include those in the area of supply chain management (SCM) or customer service, where someone in one company might use a Web interface to order items or to obtain inventory levels on a vendor's products. Information in a variety of directories may need to be furnished in order to display the correct information to a user.

DMZ See demilitarized zone.

domain or realm A domain or realm represents a single unit of security administration or trust.

DSML See Directory Services Markup Language.

Dynamic Host Configuration Protocol (DHCP) A specification for the service provided by a router, gateway, or other network device that automatically assigns TCP/IP network settings (including an IP address) to any device that requests one.

EJB See Enterprise JavaBean.

Enterprise JavaBeans (EJB) An architecture for setting up program components, written in the Java programming language, that run in the server parts of a computer network that uses the client/server model. Enterprise Java Beans is built on the JavaBeans technology for distributing program components to clients in a network. Enterprise Java Beans offer enterprises the advantage of being able to control change at the server rather than having to update each individual computer with a client whenever a new program component is changed or added. EJB components have the advantage of being reusable in multiple applications. To deploy an EJB Bean or component, it must be part of a specific application, which is called a container.

Enterprise Service Bus (ESB) is an emerging standard for integrating enterprise applications in an implementation-independent fashion, at a coarse-grained service level (leveraging the principles of service-oriented architecture) via an event-driven and XML-based messaging engine (the bus).

ESB See Enterprise Service Bus.

European Data Directive 95/46/EC Directive of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. More information can be found at http://www.cdt.org/privacy/eudirective/EU_Directive_.html

Extensible Access Control Markup Language (XACML) A standard in encoded data exchange, makes possible a simple, flexible way to express and enforce access control policies in a variety of environments, using a single language.

Extensible Markup Language (XML) A flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. For example, computer makers might agree on a standard or common way to describe the information about a computer product (processor speed, memory size, and so forth) and then describe the product information format with XML. Such a standard way of describing data would enable a user to send an intelligent agent (a program) to each computer maker's Web site, gather data, and then make a valid comparison. XML can be used by any individual or group of individuals or companies that wants to share information in a consistent way.

Extensible rights Markup Language (XrML) A machine-interpretable language, developed at Xerox PARC. It uses XML for its syntax and was previously known as DPRL. XrML is intended to be a general purpose rights language to create usage licenses or specify the rights for a digital item. XrML is a core component in enabling distribution of digital content and access to digital services such as in an e-commerce context.

Extensible Stylesheet Language (XSL) A language for creating a style sheet that describes how data sent over the Web using the XML is to be presented to the user.

Extensible Stylesheet Language Transformations (XSLT) A language used to transform XML documents into other documents. In Second Site, XSLT is used to transform XML documents into HTML tags. The XSLT standard is administered by the World Wide Web Consortium (W3C).

federation A group of two or more organizations that have agreed to allow a user from one federation partner to seamlessly access resources from another partner in a secure and trustworthy manner.

firewall A firewall is a hardware/software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets, and can block some virus attacks—as long as those viruses are coming from the Internet.

FTN Liberty Federation Termination Identification.

FULM Federated User Lifecycle Management.

Generic Security Services Application Program Interface (GSS-API) is defined in RFC 2853. GSS-API offers application programmers uniform access to security services atop a variety of underlying security mechanisms, including Kerberos.

GLBA See Gramm-Leach-Bliley Act.

Gramm-Leach-Bliley Act (GLBA) The Financial Modernization Act of 1999, also known as the "Gramm-Leach-Bliley Act" or GLB Act, includes provisions to protect consumers' personal financial information held by financial institutions. There are three principal parts to the privacy requirements: the Financial Privacy Rule, Safeguards Rule and pretexting provisions. More information can be found at:
http://www.ftc.gov/privacy/privacyinitiatives/glbact.html

Health Insurance Portability and Accountability Act (HIPAA) HIPAA is the acronym for the Health Insurance Portability and Accountability Act of 1996. The Centers for Medicare & Medicaid Services (CMS) is responsible for implementing various unrelated provisions of HIPAA, therefore HIPAA may mean different things to different people.

- HIPAA Health Insurance Reform
  Title I of the Health Insurance Portability and Accountability Act of 1996 (HIPAA) protects health insurance coverage for workers and their families when they change or lose their jobs. Visit this site to find out about pre-existing conditions and portability of health insurance coverage.
- HIPAA Administrative Simplification
  The Administrative Simplification provisions of the Health Insurance Portability and Accountability Act of 1996 (HIPAA, Title II) require the Department of Health and Human Services to establish national standards for electronic health care transactions and national identifiers for providers, health plans, and employers. It also addresses the security and privacy of health data.

More information can be found at:
http://www.cms.hhs.gov/hipaa

HIPAA See Health Insurance Portability and Accountability Act.

HTTP Point of Contact (PoC) A generic component normally located in a DMZ. It is typically an HTTP reverse proxy, or similar component, capable of authenticating a user and managing a session for that user.

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

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

identity mapping A method of creating relationships between identity properties. Some identity providers may make use of identity mapping.

identity provider (IdP) An entity that acts as a peer entity authentication service to end requestors and data origin authentication service to service providers (this is typically an extension of a security token service).

IIOP See Internet Inter-ORB Protocol.

Internet Inter-ORB Protocol (IIOP) A protocol developed by the Object Management Group (OMG) to implement CORBA solutions over the World Wide Web. IIOP enables browsers and servers to exchange integers, arrays, and more complex objects, unlike HTTP, which only supports transmission of text

intrusion defense Provides defense against attackers attempting to gain access to a network, device or host. Intrusion detection and response capabilities monitor network segments and hosts within a centralized operational and management framework. Responses to detected intrusion attempts include inputs to event management systems, paging, and trouble ticket systems. Intrusion defense is installed on hosts, desktops, mobile computers, and on-network devices. Intrusion Defense management includes the lifecycle management of intrusion detection mechanisms on hosts, desktops, and mobile computers and on network devices:

- ID application distribution and updates to authorized platforms. Host platforms may be initially loaded with ID clients or fetch ID clients from the ID manager.
- Notification that ID updates are available.
- Making ID clients and updates available for automatic download when the host platform connects to the manager.
- Receiving host ID security event logs and performance log files and host ID configuration data.
- Providing summary ID event information and alerts.
- Providing reports.

IP (Internet Protocol) address A numerical identifier for a device on a TCP/IP network. The IP address format is a string of four numbers, each from 0 to 255, separated by periods.

ITSEC Information Technology Security Evaluation Criteria, published by the European Commission.

J2EE See Java 2 Platform Enterprise Edition.

Java 2 Platform Enterprise Edition (J2EE) A Java platform designed for the mainframe-scale computing typical of large enterprises. Sun Microsystems, together with industry partners such as IBM, designed J2EE to simplify application development in a thin client-tiered environment.

Java Database Connectivity (JDBC) An application program interface (API) specification for connecting programs written in Java to the data in popular database. The application program interface lets you encode access request statements in structured query language (SQL) that are then passed to the program that manages the database. It returns the results through a similar interface. JDBC is very similar to the SQL Access Group's Open Database Connectivity (ODBC); and, with a small “bridge” program, you can use the JDBC interface to access databases through the ODBC interface.

Java Naming and Directory Interface (JNDI) Enables Java platform-based applications to access multiple naming and directory services. Part of the Java Enterprise application programming interface (API) set, JNDI makes it possible for developers to create portable applications that are enabled for a number of different naming and directory services, including file systems, directory services, such as Lightweight Directory Access Protocol (LDAP), Novell Directory Services, and Network Information System (NIS); and distributed object systems, such as the Common Object Request Broker Architecture (CORBA), Java Remote Method Invocation (RMI), and Enterprise JavaBeans (EJB).

Java Security Specific security protocols are launched to protect programs using Java, a computer programming language mostly used for the World Wide Web. Java programs, which can be downloaded from a Web server and run on Java-compatible browsers, are run in a small, constrained area called a Sandbox. The Sandbox contains a security system that checks and verifies all codes coming into it. Java Security employs data encryption, where keys are needed to encrypt and read data.

JavaServer™ Page (JSP) A technology for controlling the content or appearance of Web pages through the use of servlets, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it.

JAX-RPC A specification that describes application programming interfaces (APIs) and conventions for building Web services and Web service clients that used remote procedure calls (RPC) and XML. JAX-RPC is also known as JSR 101.

JDBC See Java Database Connectivity.

JKS Java Key Store

JNDI See Java Naming and Directory Interface.

JSP See JavaServer Page.

Kerberos A network authentication protocol developed at the Massachusetts Institute of Technology (MIT). It is designed to provide strong authentication for client/server applications across insecure network connections by using secret-key cryptography.

Key Escrow The storing of a key (or parts of a key) with a trusted party or trusted parties in case of loss or destruction of the key.

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

key recovery A process used to recover encrypted information that does not involve the storing of the key or any part of the key with a third party. Sometimes, important data needs to be recovered without normal access. The encryption key may have been lost accidentally, or an organization may need to audit its resources, or the data may be needed by law enforcement and other outside authorities. Key-recovery systems, like those proposed by National Institute for Standards and Technology (NIST), rely on close cooperation between certification authorities and user communities that share a public-key infrastructure (PKI). These groups would need to share components of encryption keys that are stored at separate locations. Many organizations find key recovery a preferable process to key escrow. The US government recently relaxed controls on the export of strong encryption based upon the development of key recovery technology by the computer industry.

LDAP See Lightweight Directory Access Protocol.

LECP Liberty-enabled Client/Proxy.

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

Lightweight Directory Access Protocol (LDAP) A software protocol for enabling anyone to locate organizations, individuals, and other resources (such as files and devices) in a network, whether on the public Internet or on a corporate intranet. LDAP is a "lightweight" (smaller amount of code) version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network.

Lightweight Third Party Authentication (LTPA) Implements an authentication protocol that uses a trusted third-party Lightweight Directory Access Protocol (LDAP) server. LTPA causes a search to be performed against the LDAP directory. LTPA supports both the basic and certificate challenge type.

LTPA See Lightweight Third Party Authentication.

MAC address Media Access Control Address a preassigned 48-bit network address that is unique to a given network interface card and can be used to identify networked devices for security purposes.

mapping rules Rules used to convert a security item from form understood by an origin process to a form understood by a destination process. For example, an application can authenticate a user via any mechanism it chooses (ID/password, certificate, and so on), and then based on the mapping rules convert the authenticated identity to an identity format defined for a directory.

MIME See Multi-Purpose Internet Mail Extensions.

Mobile Station International ISDN Number (MSISDN) The standard international telephone number used to identify a given subscriber. The number is based on the ITU-T (International Telecommunications Union-Telecommunication Standardization Sector) E.164 standard.

Multi-Purpose Internet Mail Extensions (MIME) An extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP). In 1991, Nathan Borenstein of Bellcore proposed to the IETF that SMTP be extended so that Internet (but mainly Web) clients and servers could recognize and handle other kinds of data than ASCII text. As a result, new file types were added to "mail" as a supported Internet Protocol file type.

Servers insert the MIME header at the beginning of any Web transmission. Clients use this header to select an appropriate "player" application for the type of data the header indicates. Some of these players are built into the Web client or browser (for example, all browsers come with GIF and JPEG image players as well as the ability to handle HTML files); other players may need to be downloaded.

NAT See Network Address Translation.

Network Address Translation (NAT) A security technique—generally applied by a router—that makes many different IP addresses on an internal network appear to the Internet as a single address; thus the specifics of the internal network remain hidden.

Network Information System (NIS) is a network naming and administration system for smaller networks that was developed by Sun Microsystems. NIS+ is a later version that provides additional security and other facilities. Using NIS, each host client or server computer in the system has knowledge about the entire system. A user at any host can get access to files or applications on any host in the network with a single user identification and password. NIS is similar to the Internet's domain name system (DNS) but somewhat simpler and designed for a smaller network. It is intended for use on local area networks.

network security solutions Network security solutions for on demand provide secure connectivity and access control to and for the enterprise network. Remote connections to the enterprise network can use a variety of technologies such as dialup and Virtual Private Network (SSL and IPSEC). Network firewalls permit only connections that are specified, in directions that are specified, and using protocols that are specified. Network security solutions feature centralized managed, log, and security event audit trail generation and collection, and report generation.

NIS See Network Information System.

non-repudiation Non-repudiation occurs when a document or participant in an activity is valid. In digital cryptography, this applies to a person who uses a private key to protect access. This guarantees that any messages signed using that person's digital signature could only have come from them. In e-commerce, when the key holder uses a digital signature in a financial transaction, it guarantees that the person making the transaction is who they claim to be.

OASIS See Organization for the Advancement of Structured Information Standards.

ODOE See On Demand Operating Environment.

On Demand Operating Environment (ODOE) The new computing architecture designed to help companies realize the benefits of on demand business. The on demand operating environment has four essential characteristics: It is integrated, open, virtualized, and autonomic.

Open Platform for Security Check Point (OPSEC) The initiative to provide a common architecture for integrating security solutions.

Open Systems Interconnection (OSI) A standard description or "reference model" for how messages should be transmitted between any two points in a telecommunication network. Its purpose is to guide product implementors so that their products will consistently work with other products. The reference model defines seven layers of functions that take place at each end of a communication. Although OSI is not always strictly adhered to in terms of keeping related functions together in a well-defined layer, many if not most products involved in telecommunication make an attempt to describe themselves in relation to the OSI model. It is also valuable as a single reference view of communication that furnishes everyone a common ground for education and discussion.

OPSEC See Open Platform for Security Check Point.

Organization for the Advancement of Structured Information Standards (OASIS) A global consortium that drives the development of e-business and Web service standards.

OSI See Open Systems Interconnection.

Passive Requestor An HTTP browser capable of broadly supported HTTP (for example, HTTP/1.1).

PEP Policy Enforcement Point.

PKI See Public Key Infrastructure.

Point of Contact (PoC) A generic component, normally located in the DMZ. It is typically an HTTP reverse proxy, or similar component, capable of authenticating a user and managing a session for that user. Typically the PoC will have a connection to a local user registry, used to validate user authentication credentials presented by the user and also to retrieve user attributes/privilege information used with session management for an authenticated user.

policy management Policy management in the On Demand Security Infrastructure is the consistent application of enterprise security policy to on demand infrastructure components, services, and applications; network security solutions; and on demand security infrastructure components and services. Policy management is applied independent of application logic and operating system platform and includes trusted identity and token lifecycle management identity, access control/authorization lifecycle management, federated identity lifecycle, privacy, single sign-on, compliance determination and remediation, security event auditing and processing, and failure situations.

portal A term, generally synonymous with gateway, for a World Wide Web site that is a major starting site for users when they get connected to the Web or that users tend to visit as an anchor site, linking to many other sites. Typical services offered by portal sites include a directory of Web sites, the ability to search for information, news, weather information, e-mail, stock quotes, phone and map information, and sometimes a community forum. Excite is among the first portals to offer users the ability to personalize that Web site according to individual interests.

privacy policies Security policies for managing access to and use of sensitive personal information, referred to as privacy-sensitive information. Individuals who provide personal information, such as social security numbers, have the right to determine when, how, and to what extent their personal information is used by organizations that collect the information.

profile A document that describes how this model is applied to a specific class of requestor (for example, passive or active)

proxy An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them, with possible translation, on to other servers. A proxy must interpret and, if necessary, rewrite a request message before forwarding it. Proxies are often used as client-side portals through network firewalls and as helper applications for handling requests via protocols not implemented by the user agent.

Pseudonym Service A Web service that maintains alternate identity information about principals within a trust realm or federation. The term principal, in this context, can be applied to any system entity, not just a person.

public key In asymmetric cryptography, the key that is made available for others to use to encrypt information. The owner of the associated private key is the only person who can decrypt the information.

Public Key Infrastructure (PKI) A system for verifying the authenticity of each party involved in an Internet transaction, protecting against fraud or sabotage, and for non-repudiation purposes so that consumers and retailers may protect themselves against denial of transactions. Trusted third-party organizations called certificate authorities issue digital certificates—attachments to electronic messages—that specify key components of the user's identity. During an Internet transaction signed, encrypted messages from one party to another are automatically routed to the Certificate Authority, where the certificates are verified before the transaction can proceed. PKI can be embedded in software applications, or offered as a service or a product. e-business leaders agree that PKIs are critical for transaction security and integrity, and the software industry is moving to adopt open standards for their use. Simplifying the directory systems that contain PKI data remains a challenge.

RA See Registration Authority.

RBAC See Role Based Access Control.

realm or domain Represents a single unit of security administration or trust.

Registration Authority (RA) An authority in a network that verifies user requests for a digital certificate and tells the Certificate Authority (CA) to issue it. RAs are part of a public key infrastructure (PKI), a networked system that enables companies and users to exchange information and money safely and securely. The digital certificate contains a public key that is used to encrypt and decrypt messages and digital signatures.

Remote Method Invocation (RMI) The standard specifications of the Java RPC.

RMI See Remote Method Invocation.

Role-Based Access Control (RBAC) A method of granting access rights to users based on their assignment to a defined role in the organization.

router An interconnection device that links two discrete networks and forwards packets between them. A router uses a networking protocol such as IP to address and direct data packets flowing into and out of the network on which it sits.

Sarbanes-Oxley Act The intention of the Sarbanes-Oxley Act of 2002 is to improve quality and transparency in financial reporting and independent audits and accounting services for public companies, to create a Public Company Accounting Oversight Board, to enhance the standard setting process for accounting practices, to strengthen the independence of firms that audit public companies, to increase corporate responsibility and the usefulness of corporate financial disclosure, to protect the objectivity and independence of securities analysts, to improve Securities and Exchange Commission resources and oversight, and for other purposes. More information can be found at `http://sarbanes-oxley.com`

SASL See Simple Authentication and Security Layer.

secure logging The means of recording security events and the protection provided to such logs to ensure their non-repudiation. Secure logging also includes a means for processing logs and generating reporting.

Secure Networks and Operating Systems Secure networks are networks that have implemented logical and physical access controls and may have implemented confidentiality, data integrity, and non-repudiation security services to restrict data access and network management to authorized personnel or entities. Secure operating systems are operating systems that have implemented logical and physical access controls and may have implemented confidentiality, data integrity, and non-repudiation security services to restrict data access and network management to authorized personnel or entities. Secure networks and operating systems generate security event audit records and are securely managed.

Secure Sockets Layer (SSL) A commonly used protocol for managing the security of a message transmission on the Internet. SSL has recently been succeeded by Transport Layer Security (TLS), which is based on SSL.

Security Assertion Markup Language (SAML) A specification designed to provide cross-vendor single sign-on interoperability.

Security Policy Expression® Security policy expression is the means by which security policy is applied to or implemented for specific IT system components and applications. For example, firewall filtering rules in a file, hardware settings, and network configurations.

Security Token Represents a collection of claims.

Security Token Service (STS) A Web service that issues security tokens. That is, it makes assertions based on evidence that it trusts, whoever trusts it. To communicate trust, a service requires proof, such as a security token or set of security tokens, and issues as security token with its own trust statement (note that for some security token formats this can just be a reassurance or co-signature). This forms the basis of trust brokering.

service/endpoint policy Corporate security policy applied to or developed for services and information technology endpoints including response to legal, regulatory, and legislative requirements. Service policy states the specific security requirements for a service that generally is provided by a configuration of hosts, networks components, and applications. Endpoint policy states the specific security configuration to be implemented an individual host, network component, or application, and the protocols used to implement the service policy.

service-oriented architecture (SOA) expresses a software architectural concept that defines the use of services to support the requirements of software users. In a SOA environment, nodes on a network make resources available to other participants in the network as independent services that the participants access in a standardized way.

signature A value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the signature to verify that the data has not been altered since it was signed by the signer.

Signed Security Token A security token that is asserted and cryptographically signed by a specific authority (for example, an X.509 certificate or a Kerberos ticket).

sign-in The process by which security tokens are obtained for realm/domain or federation.

sign-out The process by which security tokens are destroyed for realm/domain or federation.

Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) A mechanism that allows the secure negotiation of the mechanism to be used by two different GSS-API implementations. In essence, SPNEGO defines a universal but separate mechanism, solely for the purpose of negotiating the use of other security mechanisms. SPNEGO itself does not define or provide authentication or data protection, although it can allow negotiators to determine if the negotiation has been subverted, once a mechanism is established.

Simple Authentication and Security Layer (SASL) Defined by RFC 2222, a generic protocol framework that provides the means to use authentication mechanisms other than simple authentication and SSL over connection-based protocols. Protocols such as LDAP, POP, IMAP, and SMPT specify a SASL profile, which describes how to encapsulate SASL negotiation and SASL messages for the protocol. Within the SASL framework, different authentication schemes are referred to as mechanisms. To use SASL, a protocol includes a command for identifying and authenticating a user to a server and for optionally negotiating protection of subsequent protocol interactions. If its use is negotiated, a security layer is inserted between the protocol and the connection.

Simple Mail Transfer Protocol (SMTP) A TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, send mail is the most widely-used SMTP server for e-mail. A commercial package, sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

SMTP usually is implemented to operate over Internet port 25. An alternative to SMTP that is widely used in Europe is X.400. Many mail servers now support Extended Simple Mail Transfer Protocol (ESMTP), which allows multimedia files to be delivered as e-mail.

single sign-on (SSO) An optimization of the authentication sequence to remove the burden of repeating actions placed on the requestor. To facilitate SSO, an element called an Identity provider can act as a proxy on a requestor's behalf to provide evidence of authentication events to third parties requesting information about the requestor. These identity providers (IPs) are trusted third parties and need to be trusted by both the requestor (to maintain the requestor's identity information, as the loss of this information can result in the compromise of the requestor's identity) and the Web services that may grant access to valuable resources and information based upon the integrity of the identity information provided by the IP.

SLO Liberty Single Sign-Out.

Smart card A smart card is a small device the size of a credit card with built-in electronic memory of personal data, such as identification and financial information.

SMTP See Simple Mail Transfer Protocol.

SOA See service-oriented architecture.

SOAP A way for a program running in one kind of operating system to communicate with a program in the same or another kind of an operating system by using the HTTP Protocol and XML as the mechanisms for information exchange.

SOX See Sarbanes-Oxley Act

SP Service provider.

SPI See Stateful Packet Inspection.

SPS SSO Protocol Services.

SPNEGO See Simple and Protected GSS-API Negotiation Mechanism.

SSL See Secure Sockets Layer.

SSO See single sign-on.

Stateful Packet Inspection (SPI) A firewall technology that examines the content of packets to determine whether they will be given access to a network.

switch A hardware device that serves as a central connection point for all network cables. In a relatively small networking environment, a switch of 4 to 12 ports may be part of a router or gateway.

TLS See Transport Layer Security.

Transport Layer Security (TLS) A protocol that ensures privacy between communicating applications and their users on the Internet. When a server and client communicate, TLS ensures that no third party may eavesdrop or tamper with any messages. TLS is the successor to the Secure Sockets Layer Protocol (SSL).

trust According to the ITU-T X.509, Section 3.3.54, trust is defined as follows: "Generally an entity can be said to trust a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects".

trust domain An administered security space in which the source and target of a request can determine and agree whether particular sets of credentials from a source satisfy the relevant security policies of the target. The target may defer the trust decision to a third party, thus including the trusted third party in the Trust Domain.

trust modeling A trust model is a description/definition of how trust is established or conveyed between two entities or among multiple entities that operate under a common set of security policies.

trusted third party A mechanism in which a trusted party creates a key and then keeps a copy of it in case of loss or destruction of the key, or legitimate request from law enforcement.

UDDI See Universal Description, Discovery, and Integration.

Uniform Resource Identifier (URI) The way you identify any point of content, whether it be a page of text, a video or sound clip, a still or animated image, or a program. The most common form of URI is the Web page address, which is a particular form or subset of URI called a Uniform Resource Locator (URL)

Uniform Resource Locator (URL) The unique address for a file that is accessible on the Internet. A common way to get to a Web site is to enter the URL of its home page file in your Web browser's address line.

Universal Description, Discovery and Integration (UDDI) An XML-based registry for businesses worldwide to list themselves on the Internet. Its ultimate goal is to streamline online transactions by enabling companies to find one another on the Web and make their systems interoperable for e-commerce.

URI See Uniform Resource Identifier.

URL See Uniform Resource Locator.

Validation Service A Web service that uses the WS-Trust mechanisms to validate provided tokens and assess their level of trust (for example, claims trusted).

Virtual Organization Policies A statement of security policies for an IT system supporting the business needs of a specific subset of an enterprise or an IT system supporting cross-enterprise business needs operating under a common objective.

WAP See Wireless Application Protocol.

WAYF Where are you from.

Web services A way of providing computational capabilities using standard Internet protocols and architectural elements. For example, a database Web service would use Web browser interactions to retrieve and update data located remotely.

Web Services Description Language (WSDL) An XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically. WSDL is the cornerstone of the Universal Description, Discovery, and Integration (UDDI) initiative spearheaded by Microsoft, IBM, and Ariba.

Web Services Policy (WS-Policy) Provides a general purpose model and syntax to describe and communicate the policies of a Web service.

Web Services Security (WS-Security) Is a mechanism for incorporating security information into SOAP messages. While SOAP provides a flexible technique for structuring messages, it does not directly address how to secure these messages. WS-Security builds from the SOAP specification, structuring the use of essential security capabilities. Specifically, WS-Security uses binary tokens for authentication, digital signatures for integrity, and content-level encryption for confidentiality. By structuring SOAP security, WS-Security makes it easy to include security elements into SOAP through tools and enterprise applications.

Web Services Trust (WS-Trust) Describes a framework for trust models that enables Web services to securely interoperate.

Wireless Application Protocol (WAP) A specification for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for Internet access, including e-mail, the World Wide Web, news groups, and Internet Relay Chat (IRC). While Internet access has been possible in the past, different manufacturers have used different technologies. In the future, devices and service systems that use WAP will be able to interoperate.

Wireless Markup Language (WML) Formerly called Handheld Devices Markup Languages (HDML). A language that allows the text portions of Web pages to be presented on cellular telephones and personal digital assistants (PDAs) via wireless access. WML is part of the WAP that is being proposed by several vendors to standards bodies.

WML See Wireless Markup Language.

WSDL See Web Services Description Language.

WSP Web Services Provisioning.

X.509 A widely used specification for digital certificates that has been a recommendation of the ITU since 1988.

XACML See Extensible Access Control Markup Language.

XKMS See XML Key Management Specification.

XML See Extensible Markup Language.

XML Digital Signature (XMLDSIG) A W3C recommendation that defines an XML syntax for digital signatures. Functionally, it has much in common with PKCS#7 but is more extensible and geared towards signing XML documents. It is used by various Web technologies such as SOAP, SAML, and others.

XML encryption A process for encrypting and decrypting parts of XML documents. Most of today's encryption schemes use transport-level techniques that encrypt an entire request and response stream between a sender and receiver, offering zero visibility into contents of the interchange to intermediaries. Content-level encryption converts document fragments into illegible ciphertext, while other elements remain legible as plaintext.

XML Key Management Specification (XKMS) Leverages the Web Services framework to make it easier for developers to secure inter-application communication using public key infrastructure (PKI). XML Key Management Specification is a protocol developed by W3C which describes the distribution and registration of public keys. Services can access an XKMS compliant server in order to receive updated key information for encryption and authentication.

XMLDSIG See XML Digital Signature.

XrML See Extensible rights Markup Language.

XSL See Extensible Stylesheet Language.

XSLT See Extensible Stylesheet Language Transformations.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 1014.

- *Extending Network Management Through Firewalls*, SG24-6229
- *Tivoli Enterprise Management Across Firewalls*, SG24-5510
- *HACMP Enhanced Scalability Handbook*, SG24-5328
- *Configuring Highly Available Clusters Using HACMP 4.5*, SG24-6845
- *Understanding LDAP - Design and Implementation*, SG24-4986
- *Enterprise Business Portals with IBM Tivoli Access Manager*, SG24-6556
- *Enterprise Business Portals II with IBM Tivoli Access Manager*, SG24-6885
- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394
- *Understanding SOA Security Design and Implementation*, SG24-7310
- *Identity Management Design Guide with IBM Tivoli Identity Manager*, SG24-6996
- *Certification Study Guide: IBM Tivoli Identity Manager Version 4.6*, SG24-7118
- *Deployment Guide Series: IBM Tivoli Identity Manager*, SG24-6477
- *Deployment Guide Series: IBM Tivoli Security Compliance Manager*, SG24-6450
- *Deployment Guide Series: IBM Tivoli Access Manager for Enterprise Single Sign-On*, SG24-7350
- *Robust Data Synchronization with IBM Tivoli Directory Integrator*, SG24-6164
- *Building a Network Access Control Solution with IBM Tivoli and Cisco Systems*, SG24-6678
- *IBM WebSphere V5.0 Security WebSphere Handbook Series*, SG24-6573

- *IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series*, SG24-6198
- *WebSphere MQ Security in an Enterprise Environment*, SG24-6814
- *IBM WebSphere V5 Edge of Network Patterns*, SG24-6896
- *A Secure Portal Using WebSphere Portal V5 and Tivoli Access Manager V4.1*, SG24-6077
- *AIX 5L Version 5.2 Security Supplement*, SG24-6066
- *Using Web Services for Business Integration*, SG24-6583

Other resources

These publications are also relevant as further information sources:

- *"Technical Reference Architectures",* by Lloyd and Galambos. *IBM Systems Journal* 38, No. 1, 51–75 (1999).
- Schneider, Fred B., *Trust in Cyberspace*, National Academies Press, January 1999. ISBN 0309065585.

These publications are packaged with their corresponding software and cannot be purchased separately:

- *IBM Tivoli Access Manager for e-business Version 6.0 Release Notes,* SC32-1702
- *Tivoli Access Manager for e-business Version 6.0 Installation Guide,* SC32-1361
- *IBM Tivoli Access Manager Version 6.0 Administration Guide,* SC32-1686
- *IBM Tivoli Access Manager for e-business Version 6.0 WebSEAL Administration Guide,* SC32-1687
- *IBM Tivoli Access Manager for e-business Version 6.0 Plug-in for Web Servers Administration Guide*, SC32-1690
- *IBM Tivoli Access Manager for e-business Version 6.0 Administration C API Developer Reference,* SC32-1692.
- *IBM Tivoli Access Manager Version 6.0 Administration Java Classes Developer Reference*, SC32-1692.
- *IBM Tivoli Access Manager for e-business Version 6.0 BEA WebLogic Server Administration Guide*, SC32-1688
- *IBM Tivoli Access Manager for e-business Version 6.0 Performance Tuning Guide*, SC32-1704

- *IBM Tivoli Access Manager for e-business Version 6.0 Problem Determination Guide*, SC32-1701
- *WebSphere Application Server Network Deployment, Version 6, Securing applications and their environment*, which can be found at:
- *WebSphere Application Server, Version 6, Securing applications and their environment*, which can be found at:
- *IBM Tivoli Access Manager for Business Integration Administration Guide Version 5.1,* SC23-4831
- *IBM Tivoli Access Manager for Operating Systems Release Notes Version 6.0*, GI11-4615-00
- *IBM Tivoli Access Manager for Operating Systems Installation Guide Version 6.0,* SC23-1710
- *IBM Tivoli Access Manager for Operating Systems Administration Guide Version 6.0*, SC23-1709
- *IBM Tivoli Federated Identity Manager Release Notes Version 6.1,* GC32-1669-02
- *IBM Tivoli Federated Identity Manager Installation Guide Version 6.1.1,* GC32-1667-03
- *IBM Tivoli Federated Identity Manager Administration Guide Version 6.0,* GC32-1668-01
- *IBM Tivoli Federated Identity Manager Auditing Guide 6.1.1*, GC32-2287-01
- *IBM Tivoli Federated Identity Manager Configuration Guide Version 6.1.1*, GC32-1668-03
- *IBM Tivoli Federated Identity Manager Single Sign-on Guide Version 6.1.1*, GC32-0168-01
- *IBM Tivoli Federated Identity Manager Web Services Security Management Guide Version 6.1.1*, GC32-0169-01
- *IBM Tivoli Federated Identity Manager Problem Determination Guide Version 6.1.1*, GC32-2288-01
- *IBM Tivoli Federated Identity Manager Business Gateway Administration Guide Version 6.1.1*, SC32-1578-00
- *IBM Tivoli Federated Identity Manager Business Gateway Auditing Guide 6.1.1*, SC32-1580-00
- *IBM Tivoli Federated Identity Manager Business Gateway Problem Determination Guide Version 6.1.1*, SC32-1581-00
- *IBM Tivoli Security Compliance Manager Release Notes Version 5.1*, GI11-4695

- *IBM Tivoli Security Compliance Manager Administration Guide Version 5.1*, SC32-1594
- *IBM Tivoli Security Compliance Manager Installation Guide: All Components Version 5.1*, GC32-1592
- *IBM Tivoli Security Compliance Manager Installation Guide: Client Component Version 5.1*, GC32-1593
- *IBM Tivoli Security Operations Manager Release Notes*[1]
- *IBM Tivoli Security Operations Manager Installation Guide*
- *IBM Tivoli Security Operations Manager User Guide*
- *IBM Tivoli Security Operations Manager Administration Guide*
- *IBM Tivoli Directory Server Release Notes Version 6.0*, SC32-1682
- *IBM Tivoli Directory Server Installation and Configuration Guide Version 6.0*, SC32-1673
- *IBM Tivoli Directory Server Administration Guide Version 6.0*, SC32-1674
- *IBM Tivoli Directory Server Performance Tuning Guide Version 6.0*, SC32-1677
- *IBM Tivoli Directory Integrator 6.1.1: Reference Guide*, SC32-2566-01
- *IBM Tivoli Directory Integrator 6.1.1: Administrator Guide*, SC32-2567-01
- *IBM Tivoli Directory Integrator 6.1.1: Getting Started Guide, GI11-6480-01*
- *IBM Tivoli Directory Integrator 6.1.1: Users Guide,* SC32-2568-01
- *IBM Tivoli Directory Integrator 6.1.1: Problem Determination Guide*, SC32-2565-01
- *IBM Tivoli Identity Manager Database and Schema Reference Version 4.6,* SC32-1769
- *IBM Tivoli Identity Manager Information Center Version 4.6*, SC23-5267

  This information is available once you have installed your Identity Manager server.
- *IBM Tivoli Identity Manager Planning for Deployment Guide Version 4.6*, SC32-1708

[1] The current documentation for IBM Tivoli Security Operations Manager 3.1 is available at the following Web site:
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?toc=/com.ibm.netcool_som.doc/toc.xml

- *IBM Tivoli Identity Manager Problem Determination Guide Version 4.6*, SC32-1494
- *IBM Tivoli Identity Manager Version 4.6: Release Notes*, GI11-4212
- *Tivoli Event Integration Facility Reference Version 3.9*, SC32-1241

Online resources

The following Web sites are also relevant as further information sources:

- This RiskServer site is intended to act as a launchpad for information security and security review needs. It covers a variety of solutions and topics, including security risk analysis, information security policies, ISO 17799 (BS7799), business continuity, and data protection legislation.

  http://www.riskserver.co.uk/

- Home page for Common Criteria, which represents the outcome of a series of efforts to develop criteria for evaluation of IT security that are broadly useful within the international community.

  http://csrc.nist.gov/cc/

- CERT Coordination Center home page

  http://www.cert.org

- IBM alphaWorks Web page

  http://www.alphaworks.ibm.com

- IBM Software Categories Web page (with Security category)

  http://www.ibm.com/software/sw-bycategory/

- National Institute of Standards and Technology Computer Security Resource Center (CSRC) home page

  http://csrc.ncsl.nist.gov

- Open Group Security Forum Web page

  http://www.opengroup.org/security/topics.htm

- RFC home page

  http://www.ietf.org/rfc.html

- WebSphere product documentation Web site

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

- Tivoli product documentation Web site

  http://publib.boulder.ibm.com/tividd/td/tdmktlist.html

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

Redbooks

Enterprise Security Architecture

Using IBM Tivoli Security Solutions

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

Enterprise Security Architecture

Using IBM Tivoli Security Solutions

Audit and compliance, access control, identity management, and integrity

Extensive product architecture and component introduction

Complete coverage of Tivoli Security solutions

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

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

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

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6014-04 ISBN 0738486418