



Enterprise Security Architecture: Using IBM Tivoli Security Solutions

by Axel Bücker et al. ISBN:0738498971

IBM Redbooks © 2004 (682 pages)

This Redbook looks at the overall Tivoli Enterprise Security Architecture, focusing on the integrated identity management solution covering authentication, authorization, provisioning, and privacy, as well as risk management.

Table of Contents

Enterprise Security Architecture—Using IBM Tivoli Security Solutions

-

Preface

-

Part 1 -

Terminology and Infrastructure

Chapter 1 - Introduction

Chapter 2 - Method for Architecting Secure Solutions

Chapter 3 - IT Infrastructure Topologies and Components

Chapter 4 - Directory Technologies

Part 2 -

Managing Access Control

[Chapter 5](#) - Introduction to Access Manager Components

[Chapter 6](#) - Access Manager Web-based Architecture

[Chapter 7](#) - A Basic WebSEAL Scenario

[Chapter 8](#) - Increasing Availability and Scalability

[Chapter 9](#) - Authentication and Delegation with Access Manager

[Chapter 10](#) - Access Manager Authorization

[Chapter 11](#) - WebSphere Application Integration

[Chapter 12](#) - Access Control in a Distributed Environment

[Chapter 13](#) - Access Manager for Operating Systems

[Chapter 14](#) - Access Manager for Business Integration

[Chapter 15](#) - Tivoli Privacy Manager

Part 3 -

Managing Identities and Credentials

[Chapter 16](#) - Identity Management

[Chapter 17](#) - Introducing Directory Integrator

[Chapter 18](#) - Identity Manager Structure and Components

[Chapter 19](#) - Identity Manager Scenarios

[Chapter 20](#) - Synchronizing the Enterprise

Part 4 -

Managing a Security Audit

[Chapter 21](#) - Risk Manager Topology and Infrastructure

[Chapter 22](#) - Building a Centralized Security Audit Subsystem

[Chapter 23](#) - Extending the Centralized Security Audit Subsystem

Part 5 -

Using MASS in Business Scenarios

[Chapter 24](#) - Global MASS: An Example

Part 6 -

Appendices

[Appendix A](#) - Additional Product Information

[Appendix B](#) - Single Sign-On - A Contrarian View

[Related Publications](#)

-

[Glossary](#)

-

[Index](#)

-

List of Figures

-

List of Tables

-

List of Examples

-

Back Cover

This IBM Redbook looks at the overall Tivoli Enterprise Security Architecture, focusing on the integrated identity management solution covering authentication, authorization, provisioning, and privacy, as well as risk management throughout extensive e-business 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, Privacy Manager, Risk Manager, Directory Integrator, and Identity Manager, the Tivoli security family offers a complete set of products designed to address all of these challenges.

The book illustrates several e-business scenarios with different security challenges and requirements. It observes the IBM Method for Architecting Secure Solutions (MASS) to describe necessary architectural building blocks and components. By matching the desired Tivoli security product criteria, it describes appropriate security implementations that meet the targeted requirements.

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

Enterprise Security Architecture— Using IBM Tivoli Security Solutions

**Axel Bücker Andrew Gontarczyk Mari Heiser Santosh
Karekar Patricia Saunders Matteo Taglioni**
ibm.com/redbooks

International Technical Support Organization **Enterprise
Security Architecture Using IBM Tivoli Security
Solutions** March 2004

SG24-6014-01

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

Second Edition (March 2004) This edition applies to the following IBM Tivoli products: Access Manager for e-business 5.1, Access Manager for Business Integration 5.1, Access Manager for Operating Systems 5.1, Risk Manager 4.2, Directory Server 5.2, Directory Integrator 5.2, and Identity Manager 4.5. Various other related IBM, Tivoli and Lotus products are mentioned in this book.

© Copyright International Business Machines Corporation 2002, 2004. All rights reserved.

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

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: AIX®

alphaWorks®

DB2®

DB2 Universal Database™

Domino™

Domino Designer®

e-business on demand™

Everyplace®

IBM®

IMS™

Informix®

iSeries™

Lotus®

Lotus Enterprise Integrator®

Lotus Notes®

MQSeries®

Notes®

OS/2®

OS/390®

OS/400®

pSeries™

RACF®

Redbooks™

Redbooks (logo)™

SecureWay®

Tivoli®

Tivoli Enterprise™

Tivoli Enterprise Console®

Tivoli Management Environment®

TME®

WebSphere®

Workplace Messaging™

xSeries®

z/OS®

zSeries®

The following terms are trademarks of other companies:
Intel is a trademark of Intel Corporation in the United States,
other countries, or both.

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

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

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

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

Preface

This IBM® Redbook looks at the overall Tivoli® Enterprise Security Architecture, focusing on the integration of audit, access control, flow control, identity and credential, and integrity subsystems 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, Privacy Manager, Risk 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 and depicts several e-business scenarios with different security challenges and requirements. It uses the IBM Method for Architecting Secure Solutions (MASS) to describe necessary architectural building blocks and components. 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 IBM Method for Architecting Secure Solutions and the network topologies you will encounter when designing your infrastructure. Finally, one specific component will be explained in more detail because it belongs in every IT infrastructure today: the LDAP-based directory server.

[Part 2](#), “[Managing access control](#)” on [page 125](#) focuses on the access control subsystem of the security architecture and introduces the IBM Tivoli Access Manager components.

Part 3, “Managing identities and credentials” on [page 417](#) takes a closer look at the identity and credential subsystem with the IBM Tivoli offerings Identity Manager and Directory Integrator.

Part 4, “Managing a security audit” on [page 521](#) examines the audit subsystem and explains how the IBM Tivoli Risk Manager can be deployed effectively.

Part 5, “Using MASS in business scenarios” on [page 607](#) gives a brief example of how to apply the IBM Method for Architecting Secure Solutions in a specific customer scenario.

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

The team that wrote this redbook

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

The team that wrote this book is shown in the picture above. They are, from left to right: Patricia, Axel, Mari, Matteo, Andy, and Santosh.

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 on areas of Software Security Architecture and Network Computing Technologies. He holds a degree in computer science from the University of Bremen, Germany. He has 17 years of experience in a variety of areas related to Workstation and Systems Management, Network Computing, and e-business Solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

Andrew Gontarczyk is a Consultant with the Security and Privacy services practice in IBM Australia. He is primarily involved in security architecture design and implementation and general security consulting engagements. He has broad experience including enterprise-wide security framework design, process and policy definition, solution design and implementation, and ethical hacking. He holds a degree in Science (physics) from Macquarie University in Sydney and a degree in Electrical Engineering from Sydney University. Andrew has five years of experience within the security practice in Australia.

Mari Heiser is a senior I/T Architect with IBM in the United States, specializing in security and network architectures. She has 19 years of experience in the I/T industry related to networking, Web infrastructure, and enterprise security solutions. She holds a degree in Education from The Cleveland State University as well as a degree in Electrical Engineering. Her areas of expertise include Tivoli Access Manager, Tivoli Identity Manager, LDAP, e-business infrastructures, and networking. During the past few years, she has worked on various customer projects performing network and security assessments and architecting secure e-business infrastructures. She has written and edited several books relating to the I/T industry in general and was a contributing author of the original *Enterprise Security Architecture using IBM Tivoli Security Solutions*, SG24-6014.

Santosh Karekar is an Advisory IT Specialist - Software for IBM Global Services, India. He is primarily involved in ESM Solution architecture design and implementations and handles Project Management for Enterprise IT Security Auditing. His experience includes Enterprise Network Management, Systems Management, Process and Policy definition, and ESM solution design and implementation. He is an Engineer in Computer Technology from Bombay, India. Santosh has 13 years of industry experience, working the past three years on different Tivoli PACO & Security Modules.

Patricia Saunders is a Certified Consulting I/T Specialist with the IBM Americas Product Introduction Center, based in Dallas, TX. She currently manages beta programs for new products, which includes the development of course materials and teaching customer workshops. Pat has a Masters degree in Computing Systems Management. She has spent the past 6 years working with the Tivoli Brand in both pre-sales and services roles and has been

concentrating on the Tivoli Security product suite for the past 2 years. She has more than 20 years of experience in the IT industry.

Matteo Taglioni works as an IT Specialist in IBM Global Services Italy. He is an expert AIX® System Administrator with knowledge of WebSphere® and Tivoli products, LDAP, and directory integration. He holds a degree in Telecommunications Engineering from the University of Bologna, Italy. During the past few years, he worked on various customer projects performing e-business and security solutions design and implementation. At the moment he focuses on IBM WebSphere Portal, Tivoli Access Manager, and Tivoli Identity Manager integration.

This book is a second edition. We extend our thanks to the team who brought this great piece of information to life. Along with Mari Heiser, the first team included:

Oleg Bascurov

Cynthia Davis

Stefan Fassbender

Julien Montuelle

Rick McCarty

Guy Moins

Jim Whitmore

The following people helped kick off the first ideas and sent this project on its way:

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

Thanks to the following people for their contributions to this second release:

International Technical Support Organization, Austin Center

Betsy Thaggard

IBM US

Keys Botzum
Michael Campbell
Joseph Fitterer
Mike Garrison
Heather Hinton
Sridhar Muppidi
Ray Neucom
Benkat Raghavan
Max Rodriguez
Becky McKane
Joe Carusillo
Mark McConoughy
Shawn Young

IBM Australia

Paul Ashley
David Edwards

IBM Norway

Eddie Hartman
Johan Varno

IBM US Security Architectures

David K. Jackson
Larry Shick

IBM Germany Security Architectures

Klaus Oberhammer

IBM Switzerland

Daniel Kipfer

IBM UK Security Architectures

Phil Billin

IBM UK

Jon Harry

Avery Salmon

Vincent Cassidy

Imran Tyabji

Comments welcome

Your comments are important to us!

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

- Use the online Contact us review redbook form found at: ibm.com/redbooks
 - Send your comments in an e-mail to:
redbook@us.ibm.com
 - Mail your comments to the address on page ii.
-

Part 1: Terminology and Infrastructure

Chapter List

[Chapter 1:](#) Introduction

[Chapter 2:](#) Method for Architecting Secure Solutions

[Chapter 3:](#) IT infrastructure topologies and components

[Chapter 4:](#) Directory technologies

In this part we introduce the approach that we follow in this book and we describe the different Tivoli product areas we will take a closer look at. We also give you an introduction to the IBM Method for Architecting Secure Solutions (MASS) that will be used to develop customer security architectures.

Chapter 1: Introduction

Addressing an overall enterprise security architecture is a somewhat complex undertaking because it involves many areas. Different approaches are used in different industries, but some security textbooks that can be considered as standard. The most popular of these guides is 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.

1.1 The BS7799 security standard^[1]

The British Standard 7799 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 International Organization for Standardization 17799 (ISO 17799).

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. 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:

1. To control access to information.
2. To prevent unauthorized access to information systems.
3. To ensure the protection of network services.
4. To prevent unauthorized computer access.
5. To detect unauthorized activities.
6. To ensure information security when using mobile computing and tele-networking facilities.

3. System development and maintenance

The objectives of this section are:

1. To ensure that security is built into operational systems.
2. To prevent loss, modification, or misuse of user data in application systems.
3. To protect the confidentiality, authenticity, and integrity of information.
4. To ensure that IT projects and support activities are conducted in a secure manner.

5. To maintain the security of application system software and data.
4. Physical and environmental security
 - The objectives of this section are:
 1. To prevent unauthorized access, damage, and interference to business premises and information.
 2. To prevent loss, damage, or compromise of assets and interruption to business activities.
 3. To prevent compromise or theft of information and information processing facilities.
 - 5. Compliance
 - The objectives of this section are:
 1. To avoid breaches of any criminal or civil law; statutory, regulatory, or contractual obligations; and security requirements.
 2. To ensure compliance of systems with organizational security policies and standards.
 3. 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:

1. To reduce risks of human error, theft, fraud, or misuse of facilities.
2. 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.
3. To minimize the damage from security incidents and malfunctions and learn from such incidents.

7. Security organization

The objectives of this section are:

1. To manage information security within the company.
2. To maintain the security of organizational information processing facilities and information assets accessed by third parties.
3. 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:

1. To ensure the correct and secure operation of information-processing facilities.
2. To minimize the risk of systems failures.

3. To protect the integrity of software and information.
 4. To maintain the integrity and availability of information processing and communication.
 5. To ensure the safeguarding of information in networks and the protection of the supporting infrastructure.
 6. To prevent damage to assets and interruptions to business activities.
 7. 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.

[1]RiskServer, Security Risk Analysis, ISO17799, Information Security Policies and Business Continuity

1.2 Mitigate risk

As you saw in the [last section](#), most of the categories address different sorts of risks for an organization. It basically does not matter what kind of organization we are looking at, because everybody is facing certain risks and everybody is striving for a maximum of security.

That is why it is of immense importance that the overall enterprise security policy is the responsibility of the top business management. They have to decide where the major security risks for that type of business lie and how to proceed from there. A security policy has to define how to deal with the different categories of risks, as depicted in [Figure 1-1](#).

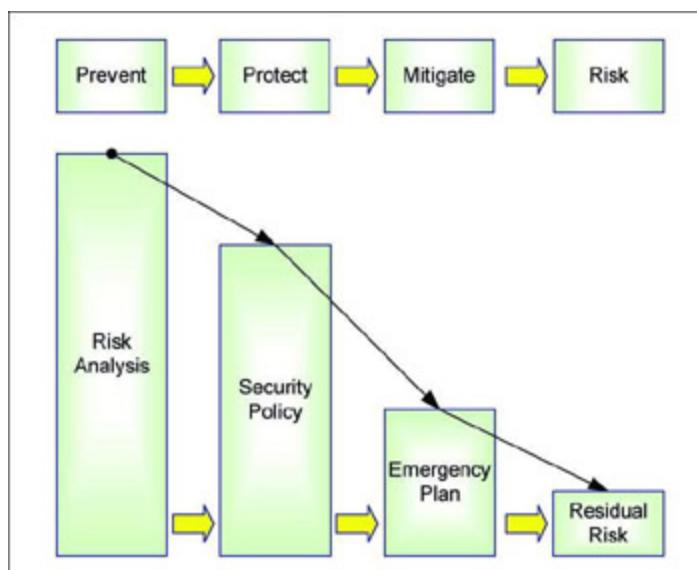


Figure 1-1: Risk categorization

1. First, you have to analyze where the major risks for your enterprise are in order to define procedures that will help prevent these risks from happening.

2. The next step is to define a security policy to deal with assets where you cannot prevent malicious actions without putting protective measures into place.
3. In case protective measures are overcome, you need to have an emergency plan that tells you what to do in those cases.
4. Because these problems are not solved simply by defining a security policy, but require investing money for certain activities and countermeasures, it is for the senior management to decide what residual risk can be accepted.

There are always insurance carriers who can provide coverage for these residual risks.

However, the depicted rate on how the risk can be reduced by defining and applying a security policy can vary in different cases.

Until now, we have purely discussed general security policies and risks without focusing on IT specifics. This will still be the case for the enterprise security policy. It includes all areas as outlined, for example, in BS7799, described in 1.1, “The BS7799 security standard” on [page 4](#). In our example in 1.3, “Corporate policy” on [page 8](#), we only focus on IT-relevant information.

Let us take a closer look at what a *Security Policy* really is by asking the question: What protection do I need?

In today’s IT, security is one of the key words. However, security can be quite complex, as complex as the organization needs or wants. While based on the same principles, it is applied differently by each enterprise, based

on business requirements. The first question should not be “do I need a firewall?” but “what do I have to protect?” or “what should I be protected from?”. A bank likely does not need the same level of security as an enterprise using the Internet to promote its image. While both require integrity of their Web site content, privacy is a specific concern for the bank. The right start is to understand the business requirements in order to anticipate the threats and address them.

1.3 Corporate policy

Technology should not drive the corporate policy; it is the other way around. When you know what you need to protect and the potential threats and risks, you can start addressing them.

First, all threats and risks will be classified in a study based on elements such as:

- Direct financial loss
- Indirect financial loss (such as investigation, recovery, and so on)
- Loss of confidential information
- Liability
- Image impact
- Cost of mitigation
- Accepting residual risk

This study can process the same threats and risks but conclude at a different severity, based on your particular business. Then the decision has to be made: accept or mitigate the risk. This process can be handled by external consultants such as IBM Global Services. The threat identification, as well as this severity study, is done in conjunction with the organization by applying a standard and a proven methodology.

It is tempting to directly translate the threat analysis into a technical solution. But first, it should lead to the corporate policy and standards. These documents highlight the risks and present how they must be handled enterprise-wide.

If it does not exist, the first document to be written should be the *corporate policy*. It must outline the high-level directions to be applied enterprise-wide. It is absolutely not technical; it is derived from the business of the enterprise and should be as static as possible, as seen in [Figure 1-2](#).

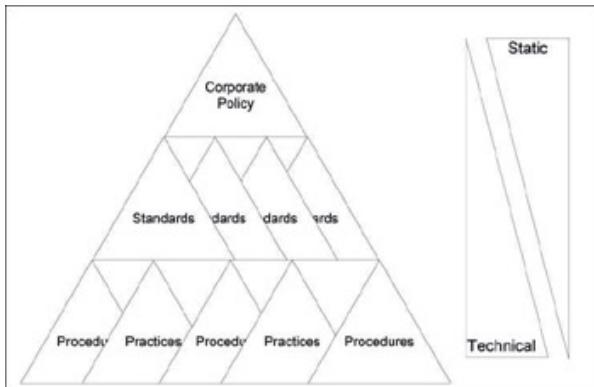


Figure 1-2: Dynamics for policy, standards, practices, and procedures

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.

1.3.1 Standards, practices, and procedures

Standards, which are derived from the corporate policy, are documents explaining how to apply the policy details in terms of *authentication*, *access control*, and so on. Changes in threats and major technology changes can have an impact on standards.

The standards are then mapped to *practices* and/or *procedures*.

Practices are descriptions of practical implementation of the standard on an operating system, application, or any other end point. They detail precise configurations, such as the services to be installed, the way to set up user accounts, or how to securely install software.

Procedures document the single steps to be applied to requests and the approval and implementation flows. A procedure could be the request to access a specific set of sensitive data, where the approval path (system owner, application owners, and so on) and conditions (Virtual Private Network [VPN], strong authentication, and so on) are explained in detail.

Tip Approval procedures are often implemented by sending e-mails or paperwork. Efficiency can be improved by using a computer to handle these repetitive tasks and ensure that changes within

the company are applied to the procedures quickly. As explained later, this can reduce human errors.

1.3.2 Practical example

Here is an example of how a policy is defined and implemented with procedures and practices.

The operations manager has reported an increased workload on the help desk due to problems caused by employees downloading non-business related programs onto their systems.

The problems range from the introduction of viruses to disruption of business processes, with a real financial impact. To address this problem, upper management incorporated the statement “the corporate assets may be used only to perform enterprise-related tasks” into the corporate policy.

First, the policy must be communicated to all employees in the enterprise.

The standards for the networking part explain which services are allowed on the employee computer. The practice then explains how to set up the Windows® or Linux clients according to the standards, and the procedures explain how to perform a request, the requirements, and the approval paths to get special services installed on your computer.

The existing clients are updated and controls are performed to verify the compliance in addition to further audit of the environment.

[Figure 1-3](#) summarizes these five steps. This is a common approach adopted in many methodologies.

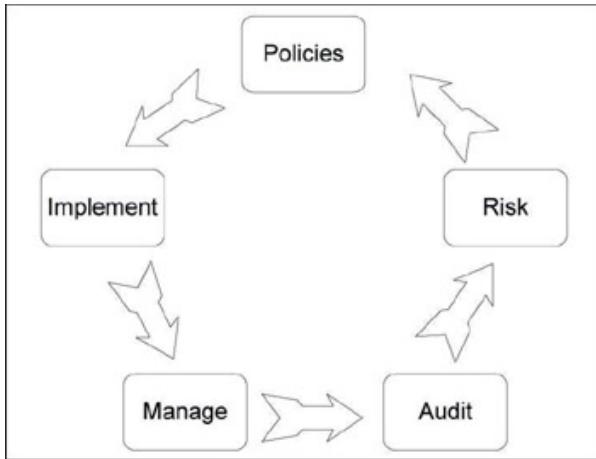


Figure 1-3: The five steps in defining your IT security

1.4 Other considerations

It is frustrating to work in an environment where you are constantly trying to discover what you can or cannot do. It is even more frustrating when services are blocked or you are being blamed without any plausible reason.

How can you accuse an employee of making unauthorized actions when it is not properly documented? Worse, if you need to take any legal action against an employee, how can you justify it without supporting documents?

As these documents will be used as rules within an enterprise, they must be clear and not subject to interpretation.

In addition, they must be publicly advertised and available. If a policy or a standard is written but not appropriately published, how can an employee be expected to follow it? It is not a loss of time to train the staff on these documents, especially when it is new and complex.

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.

The policy and the standards are 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 a legal support in case of problems.

1.4.1 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.

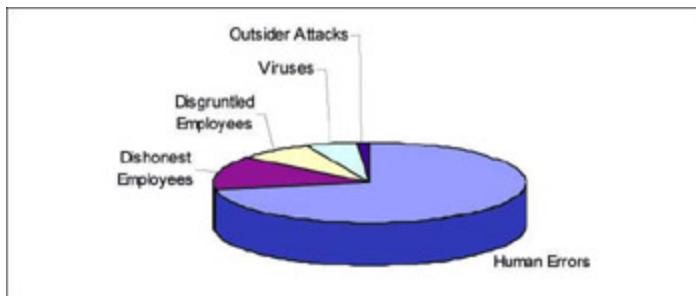


Figure 1-4: Principal threat sources

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.

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.

1.5 Closing remarks

After bringing a little more light to the general approach of developing an enterprise security architecture, and implementing a corporate policy that defines the rules of engagement when it comes to enterprise assets, we now focus on discussing the next building blocks in more detail.

In order to build a complex structural shape such as an enterprise security architecture, you need somebody 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 *Methodology for Architecting Secure Solutions*

(MASS) that reflects the current impact of thriving e-business environments. We explore this methodology in more detail in [Chapter 2, “Method for Architecting Secure Solutions”](#) on [page 15](#). We also use this methodology in several sections throughout this book in order to prove the validity of our solutions.

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. [Chapter 3, “IT infrastructure topologies and components”](#) on [page 49](#) 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 4, “Directory technologies”](#) on [page 83](#), addresses these capabilities.

Chapter 2: Method for Architecting Secure Solutions

Overview

This chapter 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].

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 chapter describes a systematic approach for defining, modeling, and documenting security functions within a structured design process in order to facilitate greater trust in the operation of resulting IT solutions.

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 life cycle 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 life cycle.

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 chapter 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 chapter, 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.

[1]Copyright 2001 International Business Machines Corporation. Reprinted with permission from IBM Systems Journal, Vol. 40, No. 3.

2.1 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.

2.2 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 3, “IT infrastructure topologies and components” on page 49](#).) IT security does not, in general, fit this model, because:

1. The security problem space has not stabilized in that the number and type of threats continue to grow and change.

2. Existing security solution frameworks take a limited view of the problem space, as with firewalls and network-level security.
3. 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.

2.2.1 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.

2.2.2 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.

2.2.3 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 to develop:

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.
-

2.3 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 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 2-1](#).

Table 2-1: Functional category Common Criteria functional class

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

Functional category	Common Criteria functional class
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.

2.3.1 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 2-1](#) on [page 23](#) 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 2-1](#) on [page 25](#) 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.

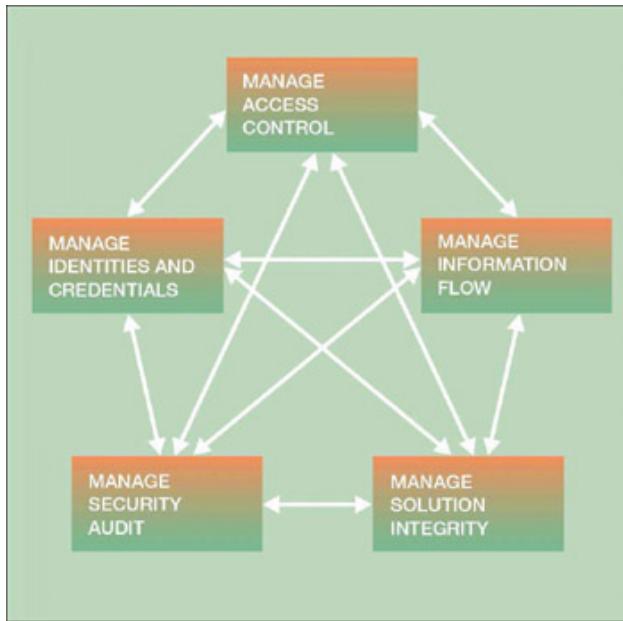


Figure 2-1: IT security processes and subsystems

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, 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 2-2](#) on [page 27](#).

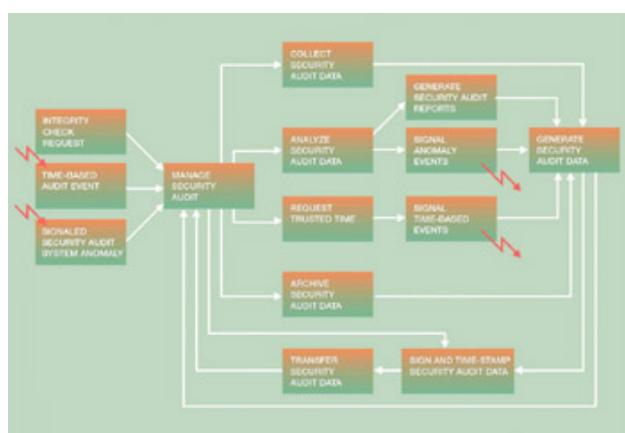


Figure 2-2: Security audit subsystem processes

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

The closed loop process for a solution integrity subsystem is represented in [Figure 2-3](#).

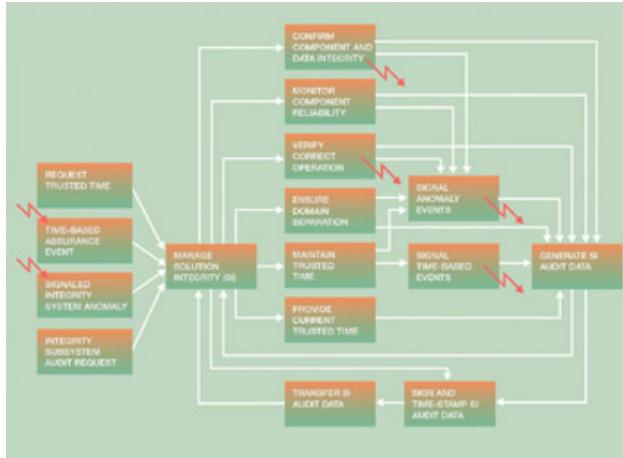


Figure 2-3: Integrity subsystem processes

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

The closed loop process for an access control subsystem is represented in [Figure 2-4](#) on [page 30](#).

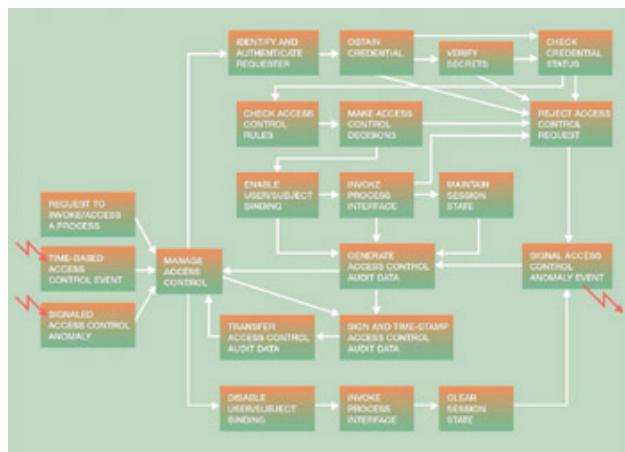


Figure 2-4: Access control subsystem processes

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

The closed loop process for an information flow control subsystem is represented in [Figure 2-5](#).

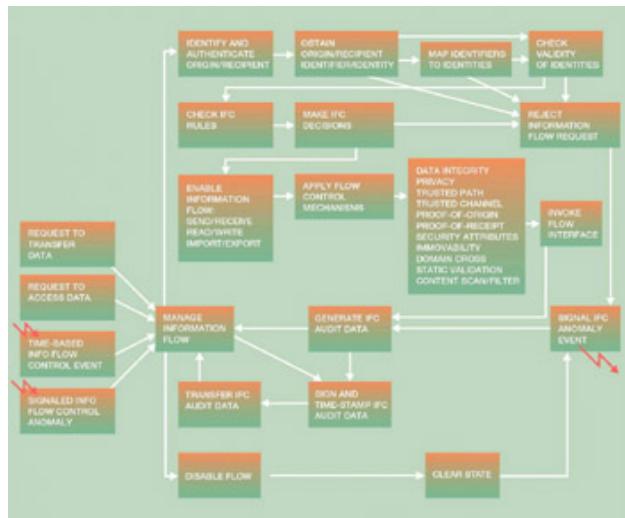


Figure 2-5: Information flow control subsystem processes

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
- Life cycle of credentials
- Anonymity and pseudonymity mechanisms

The closed loop process for a credential subsystem is represented in [Figure 2-6 on page 33](#).

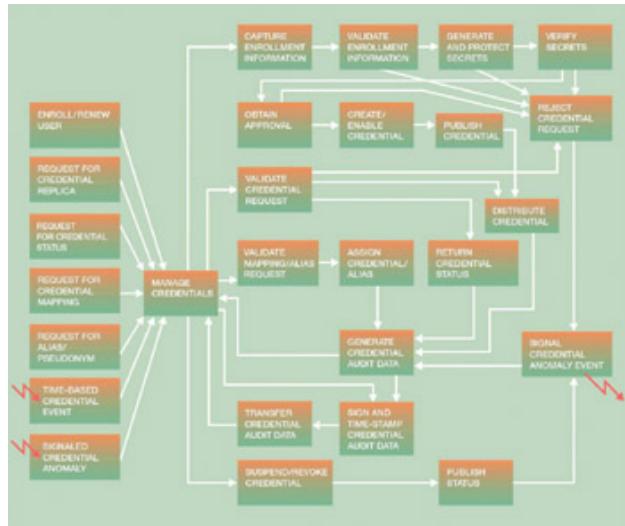


Figure 2-6: Credential subsystem processes

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.

[2]E. Rechtin, *Systems Architecting: Creating and Building Complex Systems*, Prentice Hall, 1991.

[3]Committee on Information Systems Trustworthiness, National Research Council, *Trust in Cyberspace*, National Academy Press, 1999.

2.4 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 2-7](#) 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.

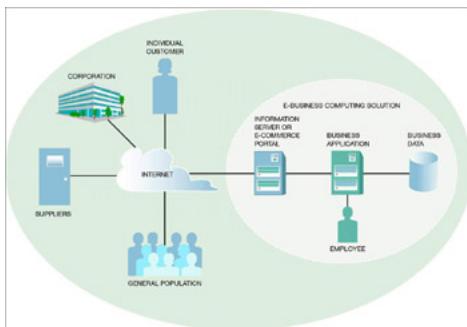


Figure 2-7: Networked information system environment
The e-business computing solution can be described as a set of automated business processes supporting the business context that requires security assurances and protections. The design goal is to infuse security into the computing solution and the related IT environment.

From a business perspective, there are two objectives:

1. To ensure that the desired IT business process flow yields correct and reliable results
2. 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.

2.4.1 Business process model

[Figure 2-8](#) 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.

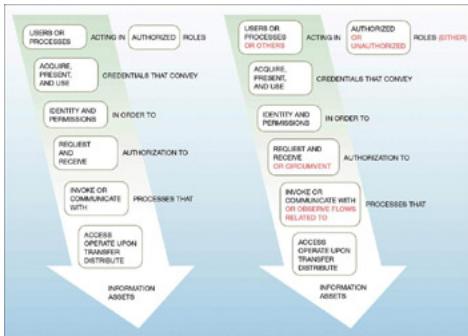


Figure 2-8: The normal and imperiled IT business process flow

2.4.2 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.

2.4.3 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 2-2](#) 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 2-2: Mapping design objectives to security subsystems

Design Objective	Subsystems Required (R)	Subsystems Supplementary (S)
1. Control access to computer systems and their processes, consistent with defined roles and responsibilities.	R	S
2. Control access to information, consistent with information classification and privacy policies.	R	S
3. Control the flow of information, consistent with information classification and privacy policies.	R	S
4. Manage the reliability and integrity of components.	R	S
5. Protect from malicious attack.	R	S
6. Ensure trusted identity for accountability of access.	R	S
7. Prevent fraud within business processes and transactions, or detect/respond to attempted fraud.	R	S

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 2-3](#) suggests motivations for instantiating security subsystems within a design. Actual subsystem enumeration requires documented rationale.

Table 2-3: 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

Subsystem	Number in a design	Characteristics of the computing environment
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

2.4.4 Documenting conceptual security architecture

Given the agreed-upon design objectives, a conceptual model for security within the IT solution can be created. Figure 2-9 on page 38 and Figure 2-10 on page 39 represent a conceptual security architecture. For clarity, security functions have been grouped by design objective.

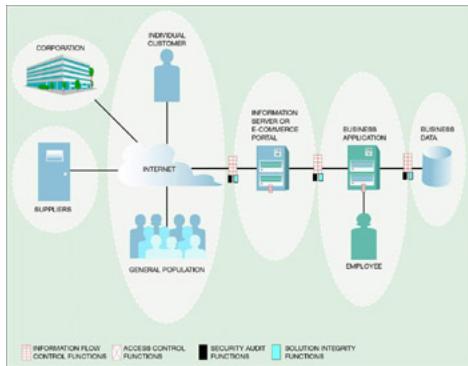


Figure 2-9: Defending against attacks

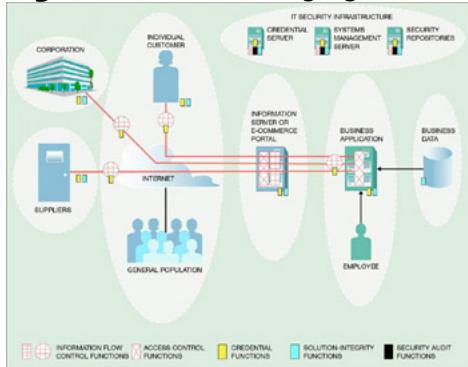


Figure 2-10: Ensuring correct and reliable operation

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, whereas an oval indicates a data flow function, such as a communication protocol with security capabilities.

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.

2.5 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.

2.5.1 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.

2.5.2 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.

Examples of architectural decisions include:

- 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

2.5.3 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 2-11 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.

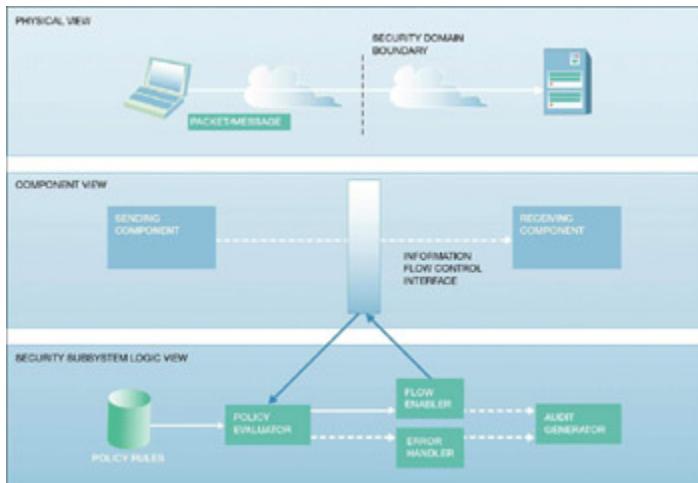


Figure 2-11: Boundary flow control with security subsystems

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.

Example 2: Three-tier client/server input flow

[Figure 2-12](#) on [page 43](#) illustrates an input flow for a three-tier client/server process that is typical of the integration of enterprise computing with the Internet environment.

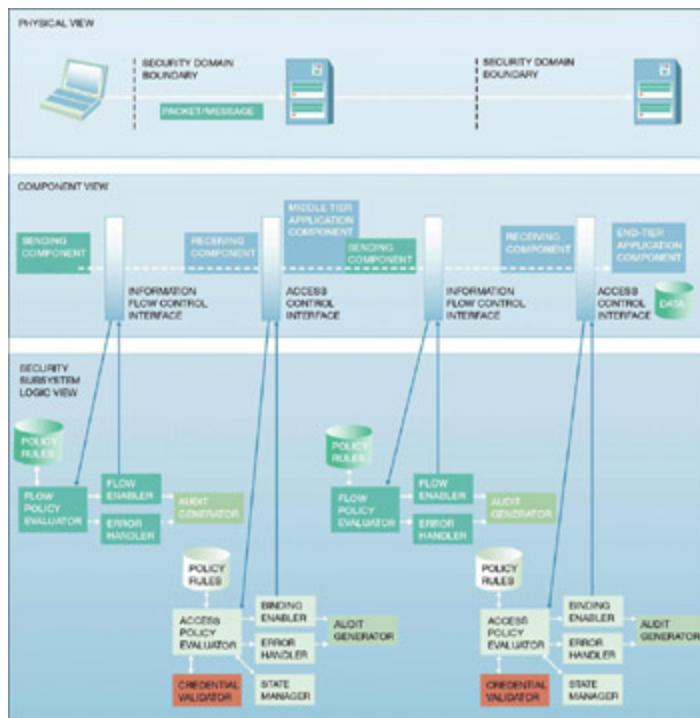


Figure 2-12: Three-tier client/server input flow with security subsystems

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.

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.

2.5.4 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 2-13](#), 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.

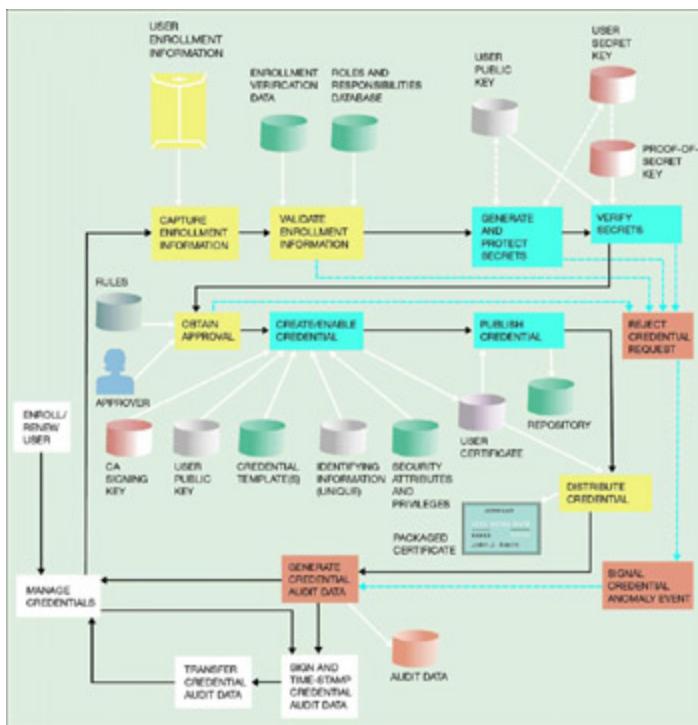


Figure 2-13: Sample PKI digital certificate enrollment process flow

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.

2.5.5 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

2.5.6 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.

[[4](#)] P. T. L. Lloyd and G. M. Galambos, “Technical Reference Architectures,” IBM Systems Journal 38, No. 1, 51–75 (1999).

2.6 Conclusions

This chapter 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.

2.6.1 Actions and further study

The concepts and the supporting detailed information presented in this chapter 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 redbook.

Chapter 3: IT Infrastructure Topologies and Components

Overview

This chapter discusses the changing role of the network. While the focus of this book is not networking or the concepts and practices used in designing or implementing them, a discussion of the basics of networking is crucial when designing or implementing a security architecture.

In addition to the network basics, we also cover some of the major application framework components in an IBM based e-business approach, such as:

- Network frameworks
 - Practical designs
 - Web servers
 - Web portal servers
 - Content servers
 - Directory technologies
-

3.1 The network becomes mission critical

Network architectures are an integral part of e-business. Networks provide the framework for making IT investment and design decisions in support of business objectives. Because the design and construction of a reliable e-business infrastructure is no longer considered only an IT issue, reliability, scalability, and flexibility are now on the minds of CEOs, CIOs, and CSOs alike.

In today's e-business setting, most companies regardless of size know that access to the Internet is essential in order to compete in the global marketplace. While the benefits of being connected to the Internet are significant, so are the risks. Connecting a private network to the Internet gives employees and business partners access to information but also supplies a pathway for external users to access a company's private information. Frequent stories in the media regarding a hacker or cracker gaining access to information via the Internet illustrate the need to implement network security.

The network and its components are the foundation on which an e-business sits. The network has become highly visible and if it and the IT structure fail, so fails the e-business. Total cost of ownership, lost revenue, lost customers, and eroded image are direct consequences. To be successful in todays global information environment, the infrastructure should be optimized to support the requirements of the business.

3.1.1 Evolution

In the beginning, network security was focused on keeping intruders out using tools such as firewalls, routers, filters,

and so on, independently installing and managing many different technologies from different vendors. The Web and its influence on todays business models has changed the assumptions and breadth of how we approach security. E-business means allowing access to someone who, in the past, would have been considered a malicious intruder into your network in a limited and careful fashion. E-business means that clients, employees, customers, and business partners alike access applications not only from the company intranet, but from the unsecured Internet as well.

The Internet was not designed with security in mind. It was constructed to be an open environment with control and trust distributed among its users, not the providers. Rules, policies, and security functions cannot be centrally administered because physical rules are impossible to apply. The firewalls, routers, filters, and so on that once secured your network are now expected to help prevent spam, viruses, worms, trojan horses, and other unwanted elements from corrupting the network and to keep thieves from stealing important company data. But they do little to keep internal employees from accessing information they are not entitled to or securing the applications that your business partners and customers use.

The adage “the best line of defense is a good offense” holds true in developing network security. Knowing the architecture of your network and defining procedures to safeguard users and data against loss and damage are the first steps. Mapping the network architecture enables you to understand the devices and pathways that applications and data take. It is the first step to implementing physical defenses against unauthorized use of resources and information.

Developing policies that enforce the overall security philosophy of the organization is the second necessary step. The focus of network security policies centers on controlling network traffic and utilization. These policies should identify the risks in the network and define acceptable use, user responsibilities, and action plans to initiate and steps to take when the policy is violated.

Analyzing the architecture and establishing points where you can create defendable boundaries to implement the policies helps to reduce the daunting task of securing the network and making security manageable. This kind of boundary is called a *perimeter network* or a demilitarized zone ([DMZ](#)).

3.2 Building network boundaries

Implementing DMZs or controlled boundaries within your network infrastructure requires the use of hardware and software devices called firewalls. A firewall utilizes a combination of hardware and software to enforce security rules that determine what information is allowed to pass into your network. 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.

Firewalls generally consist of routers, host computers, and dedicated firewall appliances. (For example, Cisco and Nokia have devices that are strictly used as firewalls.) The next few sections describe different types of firewalls and their functions.

3.2.1 Packet filter firewall

A packet filter firewall analyzes individual network packets. If a packet matches a set of rules that have been predefined in the firewall device or program, the packet filter will either allow or disallow communication based on the information in the packet and the direction it is heading. Packet filters enable you to permit or deny the transfer of data based on 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.

Packet filtering generally uses the following logic:

- If no rule is found, do not deliver.

- If a rule is found that permits the communication, the communication is allowed.
- If a rule is found that denies the communication, the communication is dropped.

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.

Packet filter firewalls often re-address the packets so that outbound traffic appears to have originated from a different host. This re-addressing is called Network Address Translation (NAT) and its primary function is to hide the trusted network from untrusted networks.

3.2.2 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.

When a connection is established, the firewall stores the following information:

- An identifier for the connection that is used for auditing purposes
- A connection state: *handshake, established, closing*
- The source IP address
- The destination IP address

- The physical network interface that the packet arrived on
- The physical network interface that the packet goes out on.

All incoming packets are compared against rules on the transport layer. These packets are either denied or accepted and passed to the network stack for delivery. It is the transport layer that maintains state information about the session in a virtual circuit table. If the packet meets all conditions of the circuit table and rules, it is transmitted up the network stack or gets sent to the destination host.

This type of firewall is more complex than packet filtering. It tracks connection information and allows for additional checks to guard against spoofing and to ensure that the data has not been modified. It also utilizes NAT to guard against advertising the trusted network to untrusted networks. Because circuit level firewalls maintain connection data about each session, it is possible to map responses back to a host.

3.2.3 Application layer firewall

Application layer firewalls examine the information in all network packets but operate in the Application Layer of the OSI model. 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.

As with circuit level firewalls, application layer firewalls utilize NAT and can perform additional checks to ensure that spoofing is not occurring. However, this ability to scan packets at the application layer comes at a price: a loss of speed. The processing time difference may seem barely noticeable, but with the speed and capability of computers today, as more connections are made to the firewall, the rate of degradation increases faster than the available bandwidth. Following that assumption, if an application layer firewall were to suffer a solid performance decline, it more likely would be related to I/O cycles required for logging and auditing than the process of stripping headers from the packet and sending it to the next application level (bit stripping).

3.2.4 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 unlike the packet filter firewall discussed earlier, it tracks each connection negotiating all interfaces of the firewall and

ensures 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.

It also offers the advantages of NAT, performs well, offers content filtering, and does not allow unsolicited UDP packets into the network. Conversely, these firewalls can be complicated to design and configure.

3.2.5 Routers

A router is 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. Simple routers for small networks can be a software package, while larger network routers are almost always a hardware appliance.

Routers are one of several types of devices that make up the backbone of a network. Hubs, LANs, switches (multipart bridges that are very simple devices used for segmenting a network into smaller pieces) and routers all take signals from computers or networks and pass them along to other computers and networks. A router is the only one of these devices that examines each packet of data as it passes and makes a decision about exactly where it should go.

One of the tools a router uses to decide where a packet should go is a *configuration table*. A configuration table is a

collection of information about which connections lead to particular groups of addresses, priorities for connections to be used, and policies for handling both routine and special cases of traffic.

It also protects the networks from one another, preventing the traffic on one from unnecessarily spilling over to the other. As the number of networks attached to one another grows, the configuration table for handling traffic among them grows, and the processing power of the router is increased.

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 3-1](#) on page 55 outlines a basic firewall installation.

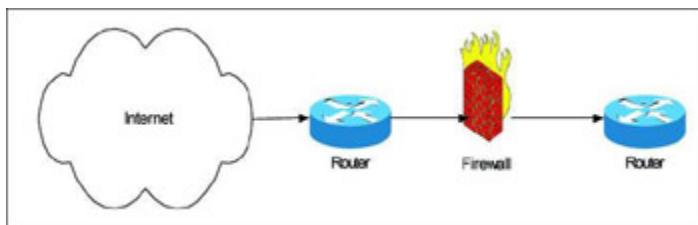


Figure 3-1: Basic Internet boundary network configuration

3.3 Connectivity framework

The best starting point is to construct a set of guiding principles to help develop the necessary infrastructure. In the past, networks evolved, meaning that as the need for a service or access to an application became necessary, the network grew to accommodate the requests. There was no unique beginning or endpoint.

Network architectures now require the input of security specialists, application developers, and network professionals. All of these individuals affect the process of the flow of data and clients on the network. Each individual brings the necessary information for assembling building blocks where the logical and physical design needs and expectations are, giving a clearer representation of the enterprise. 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 3-2](#) on page56.

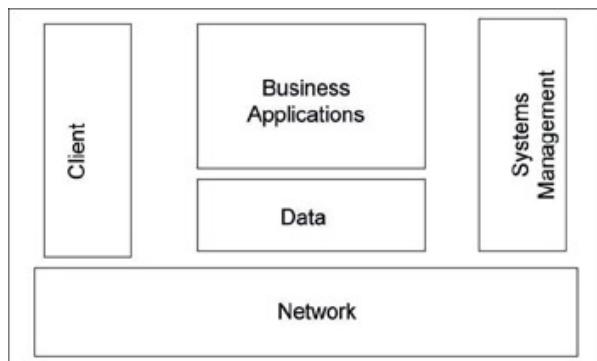


Figure 3-2: High-level architectural model including all network components

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 3-3](#).

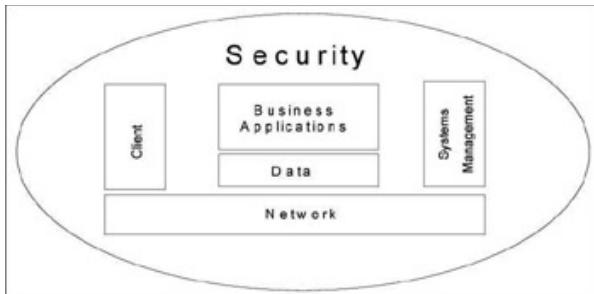


Figure 3-3: How security fits into the enterprise

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.

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.

3.3.1 Localizing a global vision with MASS

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) provides you with design objectives or, more simply put, a starting point. Based on Common Criteria, MASS is compliant with international standards that are comprehensive and well accepted. MASS provides a set of security domains to help define the threats to an enterprise (including actors/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.

Using [Figure 3-4](#), 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 3-4](#) into something more targeted, as shown in [Figure 3-6](#) on [page 63](#).

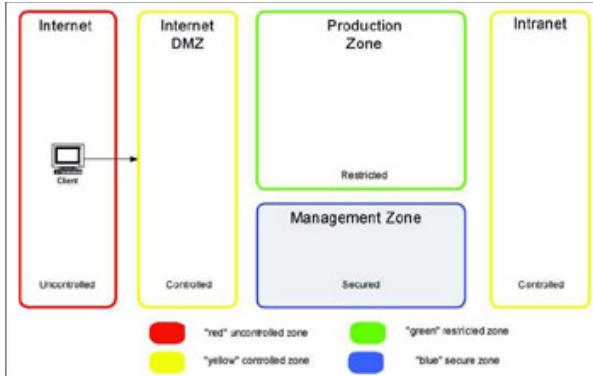


Figure 3-4: Applying MASS domain concepts Intranet

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 in this manner enables internal users to “see” out, but external users can not “see” in. 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 3-6](#) on [page 63](#) and [Figure 3-7](#) on [page 64](#) have clearly marked firewalls to help you become familiar and comfortable with the placement and domain concept.

3.3.2 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 3-5](#) on [page 62](#) summarizes the general component-type relationships and the transport classifications to the network zones discussed above.

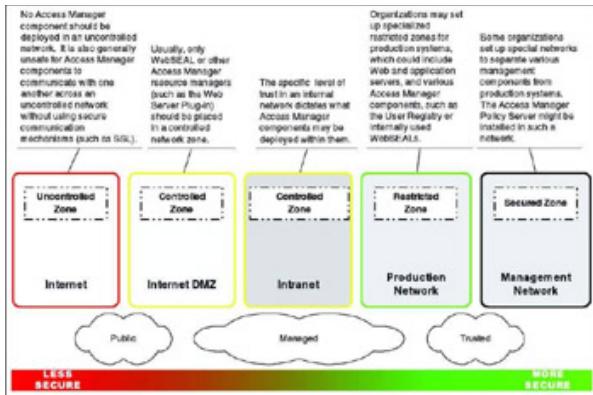


Figure 3-5: Graphic representation of network zones, transport classifications, and their level of trust

3.3.3 E-business security requirement and MASS

The IBM e-business methodology fits nicely with MASS domain concepts. MASS is built on open and accepted standards. 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 policies around your business model, not the other way around. For more information about corporate policies, see [1.3, “Corporate policy” on page 8](#).

3.4 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 3-6](#).

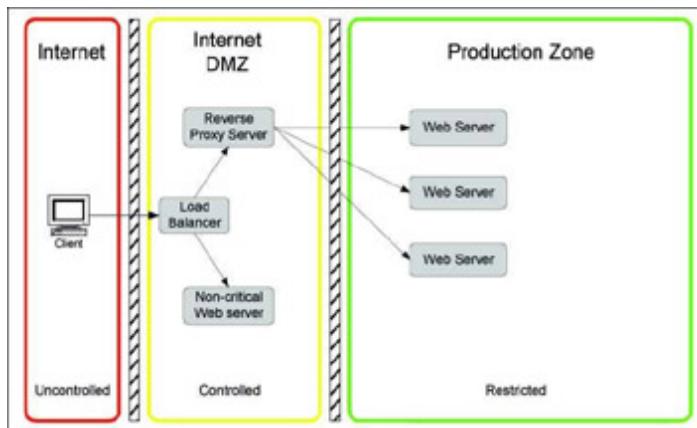


Figure 3-6: Basic DMZ design

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 3-7](#) 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.

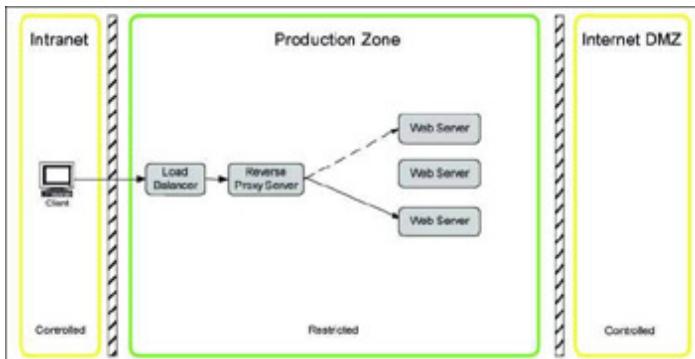


Figure 3-7: Segregating the intranet client

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.

Let us take that concept one step further: in [Figure 3-8](#) on [page 65](#), 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.

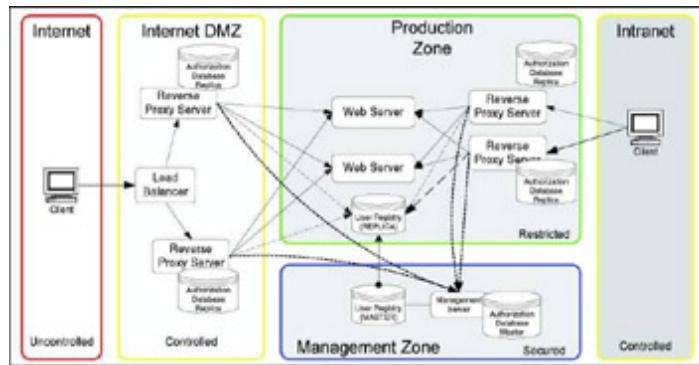


Figure 3-8: Management zone, high availability, and load balancing

3.5 Additional components

We have talked about network security structure and devices, MASS, and the zones approach to implementing security. What we have not discussed is other components involved in a Web complex.

The additional components that are discussed here were designed to work in conjunction with other IBM Tivoli security products. Software products discussed in this section are part of the IBM family of software products. However, these are not the only software solutions available and are not the only ones that IBM Tivoli security products are capable of working with.

This section touches on:

- IBM WebSphere Application Server
 - IBM WebSphere Portal Server
 - Lotus® Domino™ Product Family
 - Certificates and Certificate Authorities
 - Directory Technologies
-

3.6 WebSphere product overview

IBM WebSphere is the leading software platform for e-business on demand™. Providing comprehensive e-business leadership, WebSphere is evolving to meet the demands of companies faced with challenging business requirements such as the need for increasing operational efficiencies, strengthening customer loyalty, and integrating disparate systems.

WebSphere provides answers in today's challenging business environments and is architected to enable you to build business-critical applications for the Web. WebSphere includes a wide range of products that help you develop and serve Web applications. They are designed to make it easier for to build, deploy, and manage dynamic Web sites more productively.

In this section, we take a high-level look at the following WebSphere products:

- IBM WebSphere Studio
- IBM WebSphere Application Servers
- IBM WebSphere MQ
- IBM WebSphere Portal
- IBM WebSphere Business Integrators

3.6.1 WebSphere family

The WebSphere platform forms the foundation of a comprehensive business solutions framework; its extensive offerings are designed to solve the problems of companies of all different sizes. For example, the technologies and tools at the heart of the WebSphere platform can be used to build and deploy the core of an international financial trading application. Yet it also fits very nicely as the Web site solution for a neighborhood restaurant that simply wants to publish an online menu, hours of operation, and perhaps provide a Web-based table reservation or food delivery system. The complete and versatile nature of WebSphere can be the source of confusion for people who are trying to make important decisions about

platforms and developer toolkits for business or departmental projects. Our goal is to help you get started with understanding the technologies, tools, and offerings of the WebSphere platform.

[Figure 3-9](#) on [page 67](#) shows a high-level overview of the WebSphere platform.

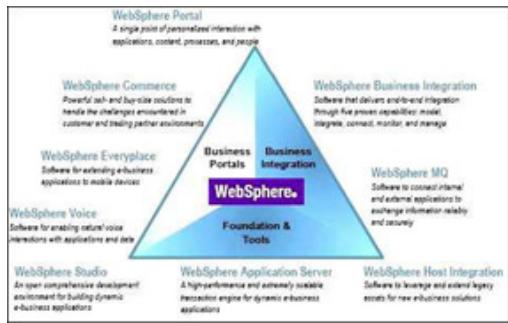


Figure 3-9: WebSphere family

3.6.2 WebSphere foundation and tools products

Foundation and tools products reduce business risk by relying on a high-quality foundation to rapidly build and deploy applications for high-performance e-business on demand. This group contains products that represent the basic functional infrastructure for other products. Key features include:

- Open services infrastructure

Provides a reliable foundation capable of high volume and secure transactions, and is fully enabled to deliver Web services. Using an open services infrastructure enables you to provide an operating environment for disparate platforms. Creating and delivering Web services means that the function only has to be developed once and can be reused anywhere it is needed. This decreases the amount of development and maintenance effort by localizing the function into one piece of code.

- Application development

Enables you to develop new applications to provide rapid and efficient responses to business needs. You can quickly build

quality applications that deliver new function and integrate with existing applications and assets.

- Enterprise modernization

Enables you to leverage existing business assets and extend them for e-business. These business assets can include applications and data that provide critical business information and processes. Incorporating these existing applications into new Web-based applications can be the quickest and most cost-effective way to move to an e-business on demand environment. Leveraging your development skills is also a key part of enterprise modernization. Creating a single, consistent, interactive development environment for your entire development organization, including Web, COBOL, and PL/I developers, can provide increased skill optimization across your development organization.

Products in this category include WebSphere Application Servers, WebSphere Studio products, and WebSphere Host Integration products.

3.6.3 WebSphere Application Server

WebSphere Application Servers are a suite of servers that implement the J2EE specification. This simply means that any Web applications that are written to the J2EE specification can be installed and deployed on any of the servers in the WebSphere Application Server family.

The foundation of the WebSphere brand is the application server, which provides the runtime environment and management tools for J2EE and Web services-based applications. WebSphere Application Servers are available in multiple configurations to meet specific business needs. They also serve as the base for other WebSphere products, such as WebSphere Commerce, by providing the application server required for running these specialized applications.

WebSphere Application Servers are available on a wide range of platforms, including UNIX-based platforms, Microsoft® operating

systems, IBM z/OS® , and iSeries™ . Although branded for iSeries, the WebSphere Application Server products for iSeries are functionally equivalent to those for the UNIX and Microsoft platforms.

WebSphere Application Server provides the environment to run your Web-enabled e-business applications. You may think of an application server as Web middleware, or a middle tier in a three-tier e-business environment. The first tier is the HTTP server that handles requests from the browser client. The third tier is the business database (for example, DB2® UDB for iSeries) and the business logic (for example, traditional business applications such as order processing). The middle tier is IBM WebSphere Application Server, which provides a framework for consistent, architected linkage between the HTTP requests and the business data and logic.

IBM WebSphere Application Server is intended for organizations that want to take advantage of the productivity, performance advantages, and portability that Java provides for dynamic Web sites. It includes:

- Java runtime support for server-side Java servlets
- Support for the Enterprise Java Beans specification
- High-performance connectors to many common back-end systems to reduce the coding effort required to link dynamic Web pages to real line-of-business data
- Application services for session and state management
- Web services that enable businesses to connect applications to other business applications, to deliver business functions to a broader set of customers and partners, to interact with marketplaces more efficiently, and to create new business models dynamically.

Configurations

Because different levels of application server capabilities are required at different times as varying e-business application scenarios are pursued, WebSphere Application Server is available in five different configurations. Although they share a common

foundation, each configuration provides unique benefits to meet the needs of applications and the infrastructure that supports them. Therefore, at least one WebSphere Application Server product configuration will fulfill the requirements of any particular project and the prerequisites of the infrastructure that supports it. The following configurations are available:

WebSphere Application Server - Express
WebSphere Application Server Network Deployment
WebSphere Application Server Enterprise
WebSphere Application Server for z/OS

As your business grows, the WebSphere Application Server family provides a migration path to higher configurations. [Figure 3-10](#) on [page 70](#) maps the progression as your requirements grow.

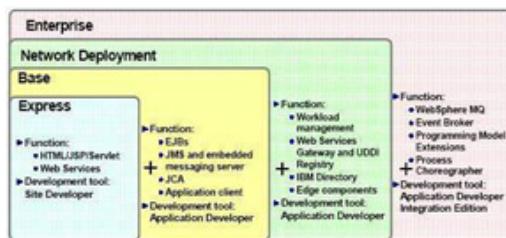


Figure 3-10: WebSphere Application Server family

WebSphere Application Server - Express

The Express configuration is a low-cost, easy-to-use, out-of-the box solution that supports simple, dynamic Web sites based on the Java servlet, Java Server Pages (JSPs), and Web Services technologies. Express enables the rapid deployment of Web applications while requiring minimal maintenance.

The Express configuration is geared to those who need to get started quickly with e-business. It is specifically targeted at medium-sized businesses and departments of a large corporation, and is focused on providing ease of use and ease of application development. It provides minimal administration overhead and one-button install, and comes bundled with a very effective development tool. Although it is easy to label WebSphere Application Server - Express as a leaner version of its more powerful siblings, it is much more than the smallest member of the family. In fact, it is an

extremely capable application server that nearly implements the full J2EE specification. It only stops short of a full implementation with respect to the Enterprise Java Bean (EJB) specification.

WebSphere Application Server - Express does not support EJB applications because a main design goal of the product was simplicity and ease of administration, and EJB applications tend to be more difficult to program and administer.

WebSphere Application Server - Express also contains enterprise-class connection frameworks that provide developers simple hooks into relational database systems. These connection frameworks are the same modules that ship with more advanced versions of the application server.

The WebSphere Application Server - Express offering is unique from the other configurations in that it is bundled with an application development tool. Although there are WebSphere Studio configurations designed to support each WebSphere Application Server configuration, they are normally ordered and installed independent of the server. WebSphere Application Server - Express provides both the server and the application development tool in a single installation.

WebSphere Application Server

The WebSphere Application Server configuration is the next level of server infrastructure in the WebSphere Application Server family. This configuration is for customers who want to use the full range of J2EE V1.3 technologies, including EJBs and JMS, but do not need workload management or central administration capabilities.

WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment is an even higher level of server infrastructure in the WebSphere Application Server family. It extends the base configuration to include clustering capabilities, edge services, and high availability for distributed configurations. These features become more important at larger enterprises where applications tend to service a larger customer base and more elaborate performance and availability requirements are in place.

Application servers in a cluster can reside on the same or multiple machines. A Web server plug-in installed in the Web server can distribute work among clustered application servers. In turn, Web containers running servlets and JSPs can distribute requests for EJBs among EJB containers in a cluster. The addition of Edge Components provides high performance and high availability features.

For example:

- The Caching Proxy intercepts data requests from a client, retrieves the requested information from the application servers, and delivers that content back to the client. It stores cachable content in a local cache before delivering it to the client. Subsequent requests for the same content are served from the local cache, which is much faster and reduces the network and application server load.
- The Load Balancer provides horizontal scalability by dispatching HTTP requests among several, identically configured Web server or application server nodes.

WebSphere Application Server - Enterprise

IBM WebSphere Application Server Enterprise provides all of the features in the Network Deployment, plus programming model extensions for complex application designs. This level of application server adds sophisticated connectors for integrating disparate and legacy data sources.

An important point to make here is that, from a technology perspective, WebSphere Application Server Enterprise V5 is very far removed from the technology base in Version 3.5. It is a set of highly valuable extensions that expand the capabilities of the core J2EE WebSphere Application Server. These extensions focus specifically on the advanced build-to-integrate capabilities and an additional set of extensions that improve the overall effectiveness of developing sophisticated applications to be run on the WebSphere Application Server platform.

It offers such capabilities as advanced application adapters, application workflow composition and choreography, extended

messaging, dynamic rules-based application adaptability, internationalization, and asynchronous processing.

Features of IBM WebSphere Application Server Enterprise include:

- Service-Oriented Architecture (SOA)

Applications need to integrate business logic and application data within the organization and outside. SOA represents all software assets as services, including legacy applications, packaged applications, J2EE components, and Web services. It also provides standards to interact between the assets. The individual assets can be reused as building blocks in other applications.

- Integrated J2EE-based workflow

Using services in a business process is the next step in application development. The integrated J2EE-based workflow offers flow-based application development. In these flows, different software assets and services can be reused and composed into a business process. The built-in engine supports human interaction and compensation for rollback-like transaction support.

- Advanced transactional connectivity

IBM WebSphere Application Server Enterprise provides numerous transaction support features essential for large enterprise applications, including dynamic application adapter support, last participant and Activity Session service support, and CORBA C++ support.

- Optimized application performance

With application profiling techniques, applications can be fine-tuned for better performance without changing a single line of code. Network Deployment provides scalability, performance, availability, and centralized management for IBM WebSphere Application Server Enterprise.

- Next generation development

WebSphere Application Server Enterprise and WebSphere Studio Application Developer Integration Edition enable next-generation development by leveraging the latest innovations that build on today's J2EE standards, providing greater control over application development, execution, and performance than was ever possible before. These include asynchronous beans, startup beans, scheduler service, and object pools.

- Increased development productivity

One way to vastly improve developer productivity is to reduce the need for handcrafted solutions that can be time-consuming, costly, and difficult to maintain. WebSphere Application Server Enterprise and WebSphere Studio Application Developer Integration Edition were designed to improve developer productivity through the use of extended messaging capabilities, internationalization service, and work areas.

- Real-time application flexibility

In order for businesses to respond quickly to changes, applications must be able to make changes and adopt to current conditions without changing the application code. Business rule beans provide a real-time framework for defining, executing, and managing business rules that encapsulate business policies. Dynamic query enables the EJB container, using the EJB Query Language, to submit queries that select, sort, join, and perform calculations on application data at runtime.

For more information about WebSphere Application Server, visit:
<http://www-3.ibm.com/software/info1/websphere/index.jsp/>

3.6.4 WebSphere Studio

WebSphere Studio is a comprehensive development environment designed to meet all development needs from state-of-the-art Web interfaces to server-side applications, from individual development to advanced team environments, from Java development to

application integration. Available in several configurations, with extensions from IBM and partners, WebSphere Studio enables developers to use a single development environment designed to meet their specific development needs.

WebSphere Studio configurations are all built on the WebSphere Studio Workbench, which extends the open-source Eclipse platform and provides an open, extensible plug-in architecture. Numerous plug-ins are available from partners and the open source community. Or, using the included plug-in development environment, you can create your own plug-ins for specific needs.

The number of experienced users of WebSphere Studio continues to increase, and WebSphere Studio has won many awards as an integrated development environment for Java, Web services, and XML.

WebSphere Studio is designed to be used by a wide variety of roles, such as Web developers, Java developers, enterprise programmers, business analysts, and system architects. From one development environment you can develop, test, deploy, and manage applications. A rich set of utilities and wizards helps to simplify common tasks so that developers can concentrate on providing true business value and on rapidly getting robust applications into production.

All products in the WebSphere Studio suite are built on open standards, and the code that they generate complies with open standards. You can build and deploy state-of-the-art, server-side applications that meet the specifications for Servlets 2.3, Java Server Pages (JSP) 1.2, and Enterprise JavaBeans (EJB) 2.0. Tools support current XML, Web services, and Unified Modeling Language (UML) standards. The open source Concurrent Versions System (CVS) is included as a source configuration management system in all products built on WebSphere Studio Workbench.

3.6.5 IBM WebSphere Portal for multi-platforms

IBM WebSphere Portal provides a single point of access to applications, application content, processes, and people in your network.

IBM WebSphere Portal enables you to establish customized portals for your employees, Business Partners, and customers. As illustrated in [Figure 3-11](#), the framework architecture implemented in this product provides a unified access point to internal and external Web applications, as well as portal access to other legacy applications. In this way, users sign on to the portal and receive personalized Web pages.

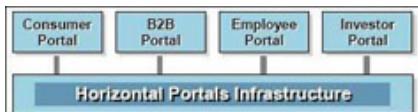


Figure 3-11: Horizontal and vertical portals

The personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases Web site usage. In addition, portals do much more; for example, they provide additional valuable functions such as security, search, collaboration, document management, document viewing, and workflow. A portal delivers integrated content and applications, plus a unified, collaborative workplace. Indeed, portals are the next-generation desktop, delivering e-business applications over the Web to all kinds of client devices.

To learn more about the WebSphere Portal product family, go to:

<http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/portals>

3.7 Lotus Domino

The Domino Application Server platform focuses on collaborative Web applications. Domino's integrated application services, such as security, workflow, and content management, optimize the platform for rapid delivery of the collaborative Web applications you need to initiate and strengthen key business relationships. Lotus software delivers one of the most comprehensive portfolios of messaging and collaboration solutions in the industry—designed to help maximize human productivity. Solutions include:

- High-performance collaboration products that help executives manage large volumes of messages and communicate in real time
- Rich e-mail and calendar products that help moderate-level users manage e-mail and calendars
- Basic messaging applications that enable low-volume users to send and receive e-mail

Domino is standards-based and offers one of the industry's most comprehensive support mechanisms for Internet messaging standards, with Internet addressing, SMTP routing, and MIME content support all native, plus full support for E/SMTP, S/MIME, SSL, POP3, IMAP4, LDAP, HTTP, HTML, SNMP, and more. Domino delivers interoperability with your current tools and systems and the ultimate in extensibility, so you can reach out to your customers and partners more easily.

Product features

IBM Lotus Domino 6.5 is a security-rich server platform that helps you and your end users work more efficiently by

letting you rapidly build collaborative applications and messaging solutions. Share information with customers, partners, and suppliers through the IBM Lotus Notes® client, a Web browser, or mobile device. Communicate using feature-rich e-mail and scheduling. Your data is always available through advanced replication and clustering features and your enterprise is protected with advanced security features. Easily upgrade to Lotus Domino 6.5 server to take advantage of new, efficiency-enhancing functions, cost-reducing administrative features, and simplified deployment and management of your infrastructure.

Lotus Domino server supports industry standards such as Simple Mail Transfer Protocol (SMTP), Multipurpose Internet Mail Extensions (MIME), Post Office Protocol (POP3), Lightweight Directory Access Protocol (LDAP), and Secure Sockets Layer (SSL). It also runs on a variety of hardware and software platforms, including Linux, so you can choose how to deploy your messaging and collaboration infrastructure without completely overhauling your environment or becoming dependent on specific hardware and software to keep your business running.

Your return on investment can increase with each collaborative solution you deploy within the Lotus Domino environment. For example, IBM Lotus Domino Designer®, an application development environment integrated with Lotus Notes software, enables developers and Web site designers to easily create, manage and deploy security-rich, collaborative applications. Developers are free to use the programming language that best fits their requirements, including the Lotus formula language, Lotus Script, Java Script, Java/CORBA, XML, Component Object Model (COM)/OLE, Messaging Application Programming Interface (MAPI), Java Server Pages (JSP) tags, and C/C++ application programming interfaces (APIs). This means that they can

use familiar languages instead of spending time and money learning a new language. Using Lotus Domino Designer 6.5, developers can enhance Lotus Domino applications with online awareness and instant messaging in a few steps, so users can interact with team members in real time to help increase their productivity.

IBM provides several ways to extend the reach of Lotus Domino data to communities beyond internal Lotus Notes client users:

- Lotus Domino server provides an integrated Web application server with browser access to data stored in both the file system and in Lotus Domino databases.
- Lotus Domino server enables developers to easily incorporate back-end enterprise data into Lotus Domino applications. Tools and services, such as IBM Lotus Enterprise Integrator® software and Lotus Domino connection services, enable connections to a company's relational databases, such as Oracle and IBM DB2 databases; enterprise resource planning systems, such as SAP, PeopleSoft, and J.D. Edwards; and transaction systems, such as Customer Information Control System servers, IBM MQSeries® software, and IMS™ transaction management system.
- With the Lotus Domino toolkit for IBM WebSphere Studio plug-in, developers can use drag-and-drop functionality to rapidly build JSP Web pages that integrate Lotus Domino data. Standard Lotus Domino portlets (for e-mail, calendar, to-do, and database views) and the more advanced Lotus Domino portlet builder enable Lotus Domino data to be easily used in IBM WebSphere Portal software.

- Lotus Domino Everyplace® software enables users to access their e-mail, calendars, to-do lists, and a company's custom Lotus Domino applications from a variety of mobile devices, including smart phones and PDAs.

3.7.1 Using two IBM application server products

For high-volume transactional applications, the optimal solution is a Web application server such as WebSphere Application Server. For collaborative, document-based applications, a superior tool is a collaborative application server (CAS) such as Lotus Domino. Throughout the IBM software group, all brands (WebSphere, Lotus, DB2, and Tivoli) have the ability to innovate and lead in a particular market, but a mandate to integrate and interoperate in ways that offer distinct customer benefits. For more information, visit: <http://www.lotus.com/>

3.8 Certificates

A digital signature is a way to ensure that an electronic document is authentic. 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 on 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. To decode an encrypted message, a computer

must use the public key provided by the originating computer and its own private key.

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. Conversely, data encrypted with the private key can be decrypted only with the public key. This asymmetry is the property that makes public key cryptography so useful.

Digital certificates encrypt data using 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.

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/>

3.9 Directory technologies

Directory servers, such as IBM Tivoli Directory Server, Novell eDirectory, Netscape Directory Server, SunOne Directory Server and Microsoft Active Directory, are a core and ever-present part of the IT infrastructure. The goal of directories is to create a single, collaborative repository that contains a complete description of the IT environment, based on a standard schema.

3.9.1 What are directories?

Directories are data stores that are similar to traditional databases; however, directories have several characteristics that distinguish them from general-purpose databases. They are intended to be logically centralized, physically distributed repositories of information. In other words, the data may be partitioned or replicated to many locations, but all clients have the same view of the data. In current usage, the information relates to users, network configuration, network applications, and policies. Information in directories is hierarchically organized (similar to an organization chart), geographically distributed, and relatively static. The goal of directories is to promote common access and management of shared data. To support shared access, directories provide application-independent data level access control (security).

Directories are not intended to be substitutes for general purpose RDBMSs. They organize data in a fundamentally different way. In directories, data is organized in a heterogeneous hierarchy. Attributes of entries can be both multi-valued and optional. Access and search mechanisms are predicated on the hierarchical structure. That is, it is easy to access and search for things that are close in terms of parent-child relationships. It is easy to perform heterogeneous searches and have heterogeneous result

sets because the type of an entry is simply an attribute. The hierarchical nature is further reinforced by the ability to distribute any subtree to a remote server, and by the security mechanism.

Characteristics of data suitable for storage in directories include:

- Data is intended to be accessed by many applications. This may imply a need for security managed by the directory.
- Access to the data may be from geographically distributed clients.
- Data requires partitioning for administration along geographic or organizational lines.
- Reads are much more common than writes.
- Data is hierarchically organized.
- Updates do not require transactions. That is, it is not necessary to perform a series of operations that must succeed or fail as a unit. This also implies that there is no need to lock access to entries or attributes.
- Information does not require strict consistency. It is acceptable for data to be temporarily out of date. For example, the consequence of a telephone number being out of date is far less severe than that of an account balance being out of date.
- Data whose individual elements are not very large. Replication can be expensive with large data elements.

3.9.2 Meta directories

In February 1996, the Burton Group defined meta directory as “the join of all the directories in the enterprise.” Since first coining this definition, the joining of corporate information has become one of the hottest topics for many of the world’s largest organizations.

As the use of directory technologies has expanded, the need to synchronize data residing in directories, databases, collaborative systems, applications, customer relationship management (CRM), Enterprise Resource Planning (ERP), and so on, has also grown. A meta directory is, simply, a directory of all directories within an organization.

This meta directory should enable the joining of many data sources into a single directory and make this data available to Web-based applications.

Meta directories are not new. However, their place within the enterprise has been ambiguous in the past. As the real cost savings of identity management in an organization become increasingly clear, as well as the need to have disparate directory systems share information, this technology will move into the forefront.

3.10 Conclusions

There are many ways to diagram an environment or implement it, as shown in the first part of this chapter. There are also many ways to keep your security and network environment safe.

Networks have been around for some time, and as have security for applications and data. Now, with the Web accessible from coffee shops as well as your corporate Internet, all of the systems that would have remained separate under the data model of the 1980s are interconnected more than before. Technologies will change, business will evolve, but the need to keep information out of the hands of those without authorization to use or see it will only increase.

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.

Chapter 4: Directory Technologies

Overview

In [Chapter 2, “Method for Architecting Secure Solutions” on page 15](#), we introduced the IBM Method for Architecting Secure Solutions and how you can design an IT security architecture based on five different subsystems addressing access control, identity and credentials, flow control, integrity, and audit. 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 section addresses these capabilities in three 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 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 last section we show the directory server developed by IBM: IBM Tivoli Directory Server.

4.1 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.

4.1.1 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 417](#) 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.

4.1.2 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.

4.1.3 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 417](#), 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 Directory Server, a Microsoft Active Directory, and a 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.

4.1.4 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.

4.2 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.

4.2.1 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.

4.2.2 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. 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, whereas 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).

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

4.2.4 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 4-1](#).

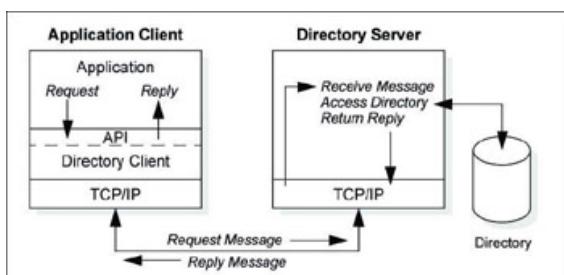


Figure 4-1: Directory client/server interaction

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.

4.2.5 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 4.2.9, “Availability and scalability” on [page 100](#).

4.2.6 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 method should only be used when data security is not an issue and when no special access control permissions are involved. No authentication is assumed when you leave the password and Distinguished Name (DN) field empty in the bind API call. The LDAP server then automatically assumes an anonymous user session and grants access with the appropriate access controls defined for this kind of access.
- Basic Authentication (BA): The client identifies itself to the server by means of a DN and a password. The server considers the client authenticated if the DN and password sent by the client matches the password for that DN stored in the directory.
- Simple Authentication and Security Layer (SASL): This is a general authentication framework, where several different authentication methods are available. Examples are Kerberos, CRAM-MD5, and EXTERNAL.

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 [3.8, “Certificates”](#) on [page 77](#) 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 4-1](#) shows an example ACL for an employee's directory entry.

Table 4-1: 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.

4.2.7 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 LDAP directory has an object class associated with it. Thus, every entry in the LDAP 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 server products that also may include some extensions to support special features they feel are common and useful to their client applications. 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 schemes 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 4-2](#) shows an example of DIT.

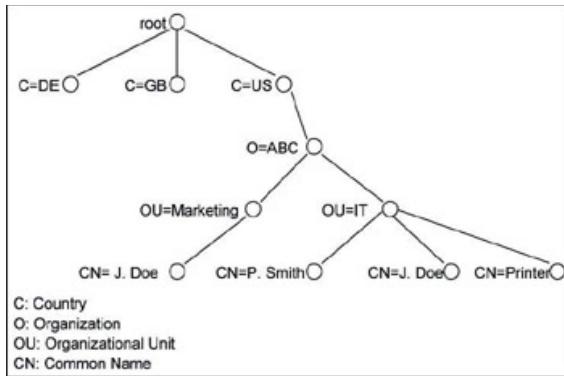


Figure 4-2: Example of a Directory Information Tree (DIT)

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=J. Doe,OU=Marketing,O=ABC,c=US` is constructed by adding the RDN `CN=J. Doe` to the DN of the ancestor entry

`OU=Marketing,O=ABC,C=US`. Note that `CN=J. Doe` is an attribute in the entry `CN=J. Doe,O=IBM,C=DE`. 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`. In the example we can see that two `CN=J. 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.

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 LDAP directory is implementation-dependent and hidden from the user of that LDAP 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.

4.2.8 Physical architecture

Although a directory server can be used for different services, as discussed in 4.2.1, “General definition” on [page 87](#), in this book we refer to it mainly

as a *user repository*.

In this section we show where to place LDAP servers with respect to security domains. Figures in this section show only the LDAP components. For a discussion of the physical network zones, refer to 3.3.1, “Localizing a global vision with MASS” on [page 57](#) and 3.3.2, “Network zones” on [page 60](#).

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 LDAP 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 4-3 on page 98](#).

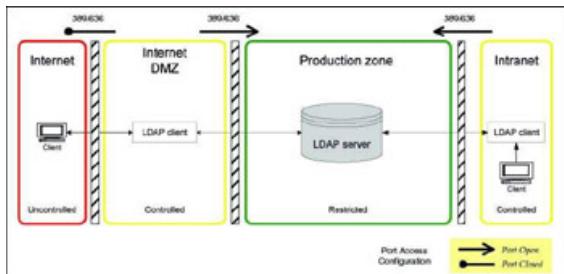


Figure 4-3: Simple architecture with one LDAP server

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 3.3.2, “Network zones” on [page 60](#), 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 4-4](#) on [page 99](#).

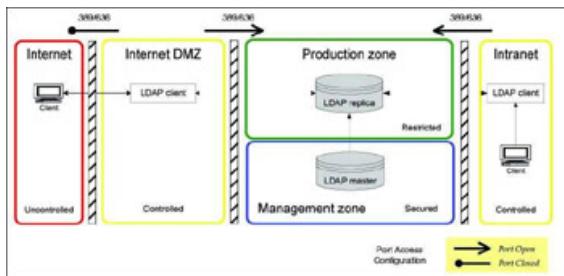


Figure 4-4: Architecture with LDAP master in a management secure domain

Both [Figure 4-3 on page 98](#) and [Figure 4-4](#) 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 95](#) 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 4-5](#) on [page 100](#). 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.

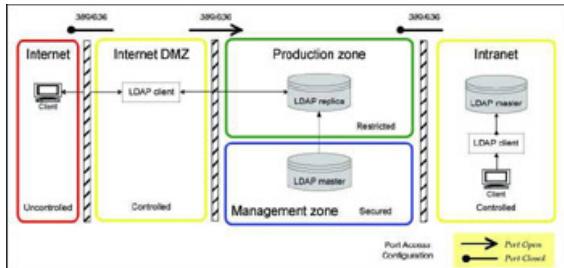


Figure 4-5: Architecture with different internal and external users repository

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 4-4](#) on [page 99](#) and [Figure 4-5](#) 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.

4.2.9 Availability and scalability

As we introduced in 4.2.5, “Distributed directories” on [page 91](#), 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 [4.3.5, “Availability and scalability” on page 114](#).

In [Figure 4-6](#) we show an example of a highly available and scalable LDAP server cluster. This figure only considers access from the Internet. For intranet access the consideration made in the [previous section](#) can be repeated.

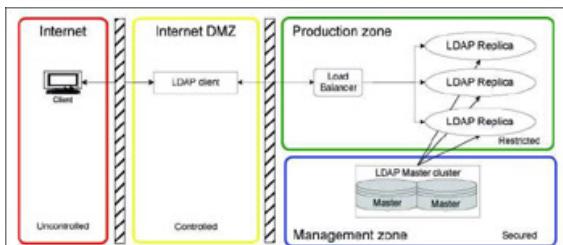


Figure 4-6: Highly available and scalable LDAP server cluster

4.2.10 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 tools features are specific to each vendor, when provided. In 4.3.6, “Logging” on [page 118](#) 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 4.2.6, “Directory security” on [page 92](#).

4.3 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 5.2, 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 Installation and Configuration Guide*, SC32-1338.

The IBM Tivoli Directory Server implements the LDAP V3 specifications as defined by the Internet Engineering Task Force (IETF). A complete overview of the supported RFCs is also listed in the *IBM Tivoli Directory Server Installation and Configuration Guide*, SC32-1338. Therefore, all general LDAP features described in [4.2, “Directories”](#) on [page 87](#) 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 product manuals, which are available online in Portable Document Format (PDF) or Hypertext Markup Language (HTML) format in the Tivoli software library at:

<http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html>

4.3.1 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 (xSeries®, 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-3.ibm.com/software/tivoli/products/directory-server/>

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

4.3.2 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 5.2 currently ships with DB2 v8.1.

- The server executable: ibmslapd.
- Tools to administer and configure the directory. These tools rely on the directory administration daemon (ibmdiradm), which runs on each server machine and also enables remote management. See [4.3.6, “Logging”](#) on [page 118](#) 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 (ldapcfg).
 - 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

LDAP applications access an LDAP 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.

4.3.3 Directory security

In 4.2.6, “Directory security” on [page 92](#), 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 LDAP 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 For server authentication, you must distribute the server certificate to each client. For server and client authentication you also must add the certificate for each client to the server's key database.

The default TCP/IP ports are those used for LDAP: 636 for secure communications and 389 for unsecure communications.

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 and as well as integrity and confidentiality.

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 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.

SASL mechanisms

Clients can also authenticate using one of the following Simple Authentication and Security Layer (SASL) mechanisms: CRAM-MD5, DIGEST-MD5, GSSAPI, and EXTERNAL. 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

User passwords stored in the directory may be encoded. This prevents clear passwords from being accessed by any users, including system administrators. The administrator may configure the server to encode *userPassword* attribute values in either a one-way encoding format or a two-way encoding format. The following four encryption options are available:

None	No encryption. Passwords are stored in the clear text format.
crypt	Passwords are encoded by the UNIX crypt encoding algorithm before they are stored in the directory.
SHA-1	Passwords are encoded by the SHA-1 encoding algorithm before they are stored in the directory.
imask	Passwords are encoded by the imask algorithm before they are stored in the directory and are retrieved as part of an entry in the original clear format. The default option is imask.

In addition to userPassword, values of the *secretKey* attribute are always *imask* encoded in the directory. Unlike userPassword, this encoding 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 must always be encoded 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 *aciEntry*. Additionally, the entry owners are the only ones who are permitted to administer the *aciEntries* 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* can not 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 *aciPropagate* 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

LDAP 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, client , 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.

4.3.4 Schema

A schema is a set of rules that governs the way that data can be stored in the directory. As we described in 4.2.7, “Schema and namespace” on [page 94](#), 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.

4.3.5 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 4.2.9, “Availability and scalability” on [page 100](#).

IBM Tivoli Directory Server allows several replication topologies that can fit different requirements. These topologies include:

- Master - Replica topology
- Master - Forwarder - Replica topology
- Peer replication topology
- Gateway topology

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

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 simple topology with four replicas is shown in [Figure 4-7](#) on [page 115](#).

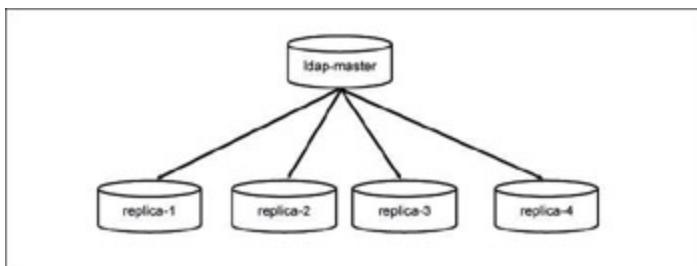


Figure 4-7: Master-replica topology

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

- SASL EXTERNAL mechanism with SSL
- Kerberos authentication

Refer to 4.3.3, “Directory security” on [page 106](#) for explanations about these methods.

Master - forwarder - replica

This topology adds one element to the previous one. In addition to the three benefits pointed out for a master - replica topology, the introduction of forwarders enables retrieval of the replication workload from the master servers. This is particularly useful in a network that contains many widely dispersed replicas.

A forwarding or cascading server is a replica server that replicates all changes sent to it. The use of forwarder 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 4-8](#).

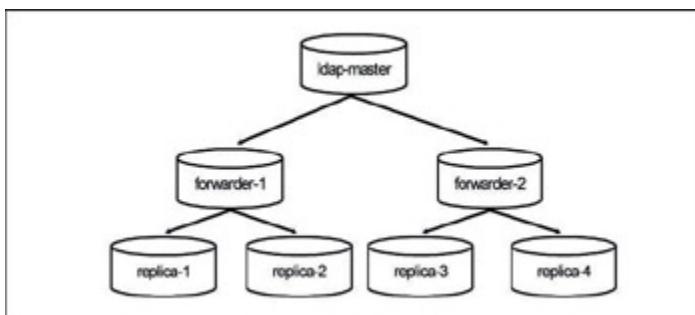


Figure 4-8: Cascading replication

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 4-9 on page 117](#).

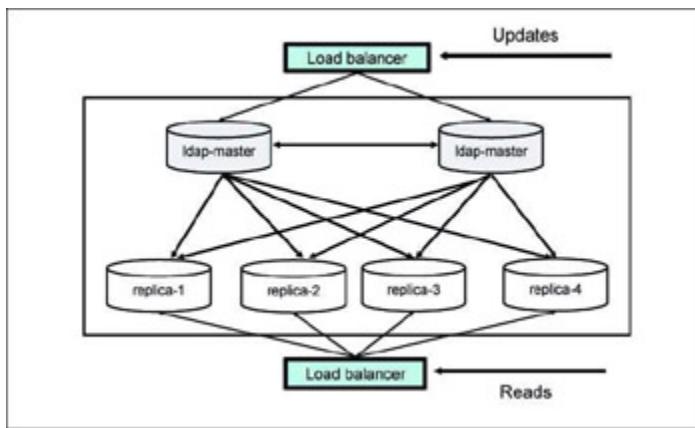


Figure 4-9: Peer replication

Peer and forwarding replication can be combined. For example, two forwarder servers could be added to the topology shown in [Figure 4-9](#).

Attention Updates to the same entry made by multiple servers might cause inconsistencies in directory data because there is no conflict

resolution. Unlike a multi-master environment, LDAP servers accept the updates provided by peer servers and update their own copies of the data. No consideration is given for the order in which the updates are received or whether multiple updates conflict. Use peer replication only in environments where the update vectors are well known. This is intended to prevent the scenario of one server deleting an object, followed by another server trying to modify the object.

Gateway topology

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 4-10](#) shows a gateway topology with four sites and four gateway servers.

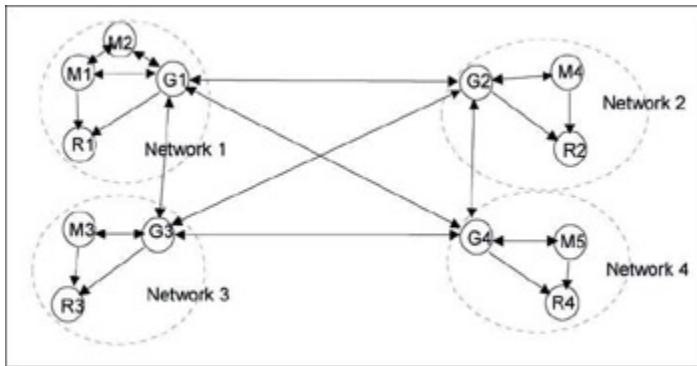


Figure 4-10: Gateway replication M4

4.3.6 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
- 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 [Chapter 17, “Introducing Directory Integrator”](#) on [page 441](#)) uses the Changelog to individuate the data modifications that have to propagate to other repositories.

4.3.7 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 4.3.2, “Base components” on [page 105](#), the two main tools are the Web Administration Tool graphical user interface (GUI) and the command line utilities. In

addition, the *ldapxcfg* 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 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 4.3.3, “Directory security” on [page 106](#), 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, including:
 - Starting and stopping the server
 - Checking server status
 - Managing server connections
 - Managing connection properties
 - Creating, managing, and removing an administrative group
 - Creating, managing, and removing unique attributes
- Setting server properties, including:
 - Changing server ports and enabling language tags
 - Setting performance
 - Setting and controlling searches
 - Enabling and disabling transaction support
 - Enabling and disabling event notification
 - Adding and removing suffixes
 - Creating and removing referrals
- Configuring security settings, including:
 - Configuring TLS and SSL

- Setting the level of encryption
- Setting password encryption
- Setting password policy
- Setting password lockout
- Setting Kerberos
- Setting certificate revocation verification
- Configuring the DIGEST-MD5 mechanism
- Managing the IBM Directory schema, including:
 - Managing object classes and attributes
- Managing replication, including:
 - Creating and modifying replication topology and replication agreements
 - Monitoring replication status
- Managing logs, including:
 - Viewing error, DB2, and administration daemon error logs
 - Modifying the error, DB2, and administration daemon logging settings
 - Viewing, enabling, and disabling the directory and administration daemon audit logs
- Managing directory entries, including:
 - Browsing the tree

- Adding, copying, modifying, and deleting an entry
- Managing language tags
- Adding or deleting auxiliary object class
- Changing group membership
- Searching the directory entries with or without filters
- Managing Access Control Lists, including performing all functions described in previous sections
- Managing group, roles, and proxy authorization group
- Performing user-specific tasks, including managing realms, templates, groups, and users

These same tasks can be executed using the Managing Entry panel, but administrators could find it easier and faster to use these user-specific functions.

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. Finally, some tasks are available

with command line utilities, but not on the Web Administration Tool (such as the `ldapdiff` utility).

The server command line utilities include the following executable programs:

- `ibmdiradm` to start the administration daemon.
- `ibmdirctl` to start and stop LDAP server.
- `bulkload`, `dbback`, and `dbrestore` to manage data backup.
- `b2ldif` and `ldif2db` to import and export data using LDAP Data Interchange Format (LDIF).
- `ldif` to convert arbitrary data values to LDIF.
- `ldapdiff` to compare data between different directories. This is useful to verify whether master and replica are synchronized.
- `ldaptrace` to dynamically activate tracing of the directory server.

Client command line utilities are available to manage entries. These include:

- `ldapmodrdn` to modify relative distinguished name
- `ldapdelete` to delete an entry
- `ldapmodify` to modify an entry
- `ldapsearch` to search an entry
- `ldapadd` to modify an entry
- `ldapchangepwd` to change the password

These commands can be combined (for example, in shell or perl scripts) to generate more complex and powerful programs.

4.3.8 Conclusion

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 fits 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. [Part 3, “Managing identities and credentials”](#) on [page 417](#), addresses this topic in more detail.

Part 2: Managing Access Control

Chapter List

[Chapter 5:](#) Introduction to Access Manager Components

[Chapter 6:](#) Access Manager Web-based Architecture

[Chapter 7:](#) A Basic WebSEAL Scenario

[Chapter 8:](#) Increasing Availability and Scalability

[Chapter 9:](#) Authentication and Delegation with Access Manager

[Chapter 10:](#) Access Manager Authorization

[Chapter 11:](#) WebSphere Application Integration

[Chapter 12:](#) Access Control in a Distributed Environment

[Chapter 13:](#) Access Manager for Operating Systems

[Chapter 14:](#) Access Manager for Business Integration

[Chapter 15:](#) Tivoli Privacy Manager

In this part, we discuss the solutions Tivoli offers in the access control subsystem of the overall security architecture. Access control information, which generally evolves around authentication and authorization mechanisms, is mainly handled 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.

Chapter 5: Introduction to Access Manager Components

Overview

In this chapter we introduce at 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 components of IBM Tivoli Access Manager, which forms the basis of Tivoli security products.

The following products make up the Access Manager family:

- Access Manager for e-business
- Access Manager for Business Integration
- Access Manager for Operating Systems

Another product related to managing access control from the privacy realm is:

- IBM Tivoli Privacy Manager

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 is discussed in [Chapter 9, “Authentication and delegation with Access Manager” on page 235](#).

- 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 legacy 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 263](#).

Some existing Access Manager resource managers include:

- IBM Tivoli Access Manager WebSEAL

WebSEAL manages and protects Web-based information and resources. WebSEAL is included with Tivoli Access Manager for e-business.

- IBM Tivoli Access Manager for Operating Systems

Access Manager for Operating Systems provides a layer of authorization policy enforcement on 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 13, “Access Manager for Operating Systems” on page 339](#) 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 367](#).

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. Later, we explain how the control of access privileges is expansive, with WebSEAL or the Access Manager Plug-in for Web Servers component able to manage access control to Web servers, and with advanced Java-based capabilities to manage access control at the Java 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. 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 addition to authorization policy enforcement, mechanisms are provided to verify defined policy and audit authorization decisions.

5.1.3 Access Manager for Business Integration

IBM 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 MQSeries and IBM WebSphere MQ products. All subsequent general references refer to IBM WebSphere MQ.

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.

5.1.4 Tivoli Privacy Manager

IBM Tivoli Privacy Manager for e-business is an enterprise privacy management solution designed to provide middleware to abstract privacy and data-handling rules from applications and IT systems. With this approach, privacy-sensitive data is linked to policy at the point of collection, and subsequent requests to use the data are then filtered (and permitted or denied) according to policy and the data owner's preferences. Audit trails of data usage can also be generated automatically. Tivoli Privacy Manager for e-business enables organizations to manage personal information in an automated manner that can help cut the costs of privacy management and mitigate the risks of unauthorized disclosure.

Tivoli Privacy Manager:

- Enforces privacy policies across your IT infrastructure
- Converts privacy policy from prose to Platform for Privacy Preferences (P3P) format
- Provides an easy-to-use natural language interface to author and manage your policy
- Monitors access to personal information and generates detailed audit logs
- Manages notification and consent preferences for information sharing across your enterprise
- Automatically generates reports detailing compliance to corporate policies

More information about Tivoli Privacy Manager can be found in [Chapter 15, “Tivoli Privacy Manager”](#) on [page 393](#).

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 Methodology for Architecting Secure Solutions (MASS) is 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 Risk Manager, the Tivoli security products provide the access control, identity management, and threat management capabilities required for any enterprise.

5.2.2 MASS subsystems

Methodology for Architecting Secure Solutions (MASS) uses the concept of architectural subsystems 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.

Access Manager is primarily an access control solution. However, the main MASS subsystems addressed by Access Manager 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 any threat management system.

Access Manager utilizes all the MASS subsystems, but these two are fundamental to the Access Manager-to-MASS mapping and the position of Access Manager within an overall Enterprise Architecture.

From an Enterprise Architecture perspective, other products will be required to satisfy all 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 136](#) 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.

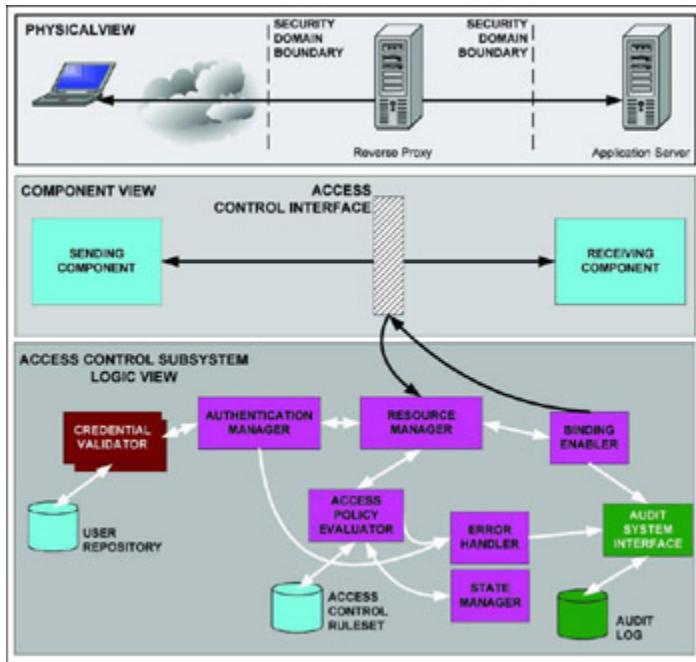


Figure 5-1: Access control subsystem

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.

Core 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 Proxy Policy 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 (roles) 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 an Extended Privilege Attribute Certificate (EPAC).

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. WebSEAL maps the user name and group information to a common domain-wide representation and format known as the Extended Privilege Attribute Certificate (EPAC).

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.

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
- Netscape iPlanet Directory
- Microsoft Active Directory (Windows 2000 advanced server only)
- Lotus Domino
- OS/390® Security Server
- z/OS Security Server LDAP Server

The IBM Directory Server is included with Access Manager and is the default LDAP directory for implementing the user registry.

Access Manager components support the use of directory replicas, and in production installations, it is generally recommended that you use replicas for scalability and availability purposes.

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 entries (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

Authorization rules are defined 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 5-2 on page 142](#) depicts the relationships between the protected object space, ACLs, and POPs. The different security mechanisms are discussed in more details in [Chapter 10, “Access Manager authorization” on page 263](#).

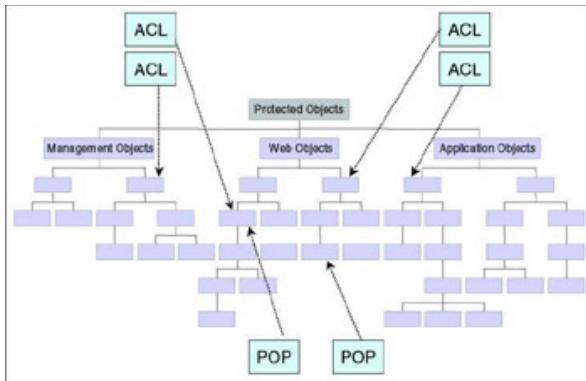


Figure 5-2: Relationship between the protected object space, ACLs, and POPs

Successful implementation of a security policy requires that the different content types are logically organized (and that the appropriate ACL and POP policies are applied). Access control management is simplified by structuring the protected resources in such a way as to minimize the number of ACL and POP 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.

There can only be a single Policy Server in an Access Manager domain. 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 217](#).

5.3.5 Proxy Policy Server

The Proxy Policy Server enables an architecture to be created where the only incoming SSL sessions to the Policy Server come from the Proxy Policy Server. This facilitates increased security because a firewall protecting the Policy Server only has to allow inbound connections from the physical Proxy Policy Server, rather than from all Access Manager servers. The SSL session from Access Manager applications to the Proxy Policy Server is independent from the SSL session from the Proxy Policy Server to the Policy Server.

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 161](#). 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 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 (WPM) provides a browser-based graphical user interface (GUI) for Access Manager administration.

It replaces the Management Console used in earlier releases of Access Manager.

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 5.4.1, “WebSEAL” on [page 148](#)) 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.

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 (for example, managing WebSEAL junctions must be done using pdadmin), and the command line interface will still be required in certain cases.

The Web Portal Manager also provides a delegated user administration capability. This enables an Access Manager administrator to create delegated user domains and assign delegate administrators to these domains, as shown in [Figure 5-3](#) on [page 145](#).

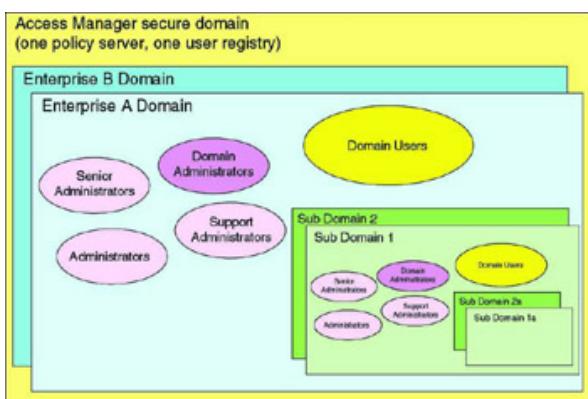


Figure 5-3: Access Manager delegation model

The initial aim of the Web Portal Manager 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.

The WPM provides a simple Web-based interface to the existing delegated administration function of Access Manager. It enables a super user (for example, sec_master) to define a number of Enterprise domains, each with one or more domain administrators that are the super users in that domain.

A Domain Administrator can create, modify, and delete domain users within the domain and can delegate administration within the domain. Domain administrators can also create subdomains inside the domain they control.

All administrative users in a domain (including the Domain Administrators) can be limited so they can only view or modify the users within their domain.

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	

Action/role	Domain admin	Senior admin	Admin	Support	Any other
Add existing Access Manager user as an administrator	X	X	X		
Create domain user	X	X			
Remove user	X	X			
Domain control	X				

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 5.0.2 is provided with Access Manager to support this capability. [Figure 5-4](#) on [page 147](#) provides an architectural view of how the Web Portal Manager works.

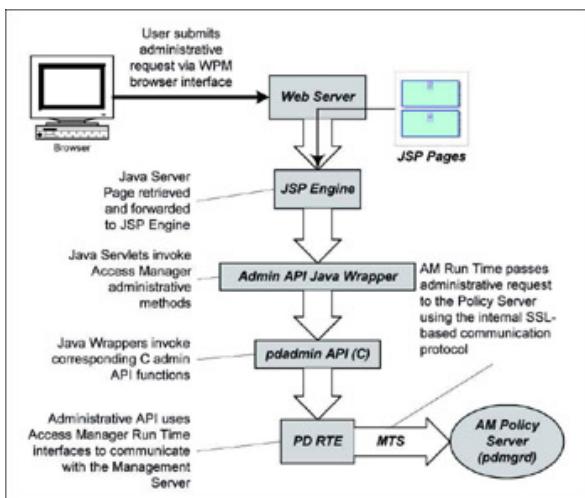


Figure 5-4: Web Portal Manager architecture

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.

5.4.1 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 140](#)). WebSEAL can provide single sign-on 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 [Chapter 6, “Access Manager Web-based architecture”](#) on [page 165](#).

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.

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.

It was hinted above that more than one target server may be defined for a particular 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. 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*.

This assumes that processing a request does not involve any security considerations. While WebSEAL is capable of doing a fine job of simply managing access to Web-based content and applications via simple junctions, this, of course, leaves out a primary purpose of utilizing WebSEAL. Its main reason for existence is integration with the base Access Manager services to provide access control and flexible authentication services for Web resources.

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 is 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.

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 HTTP authentication challenge to the browser (it supports multiple authentication mechanisms, which are discussed in 9.4, “Supported WebSEAL authentication mechanisms” on [page 246](#)). 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 from inactivity. Subsequent URL requests for this session continue to be checked to determine whether access is permitted. In this manner, WebSEAL provides single sign-on capabilities for Web-based content and applications.

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.

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.

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 an 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.

As an added functionality, WebSEAL supports *forced login* and *switch user* functions. Forced login is used to force a user to log in again, based on a policy setting or a session time-out. Switch user 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.

Delegated user authentication

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 an Lightweight Third-Party Authentication (LTPA) cookie identifying the user into the HTTP stream that is passed to the junctioned server.

WebSEAL-delegated authentication capabilities are discussed more fully in 9.5, “WebSEAL delegation mechanisms” on [page 253](#).

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, because each WebSEAL replica maintains active session states for its own authenticated users, the load balancing mechanism used must support a “sticky” capability to route subsequent requests to the same replica. (In other words, users log on to a specific WebSEAL server.) The use of WebSEAL replicas is discussed and illustrated in [Chapter 8, “Increasing availability and scalability”](#) on [page 215](#).

Virtual hosting

Multiple instances of WebSEAL can be created on a single machine using the WebSEAL configuration/unconfiguration utility. Each WebSEAL requires its own unique IP/port combination. This

approach enables something similar to a virtual hosting solution by running multiple WebSEALs with unique URL identifiers.

Communication protocols

WebSEAL can communicate with the clients and back-end servers with both encrypted (SSL) and unencrypted (TCP) protocols. The supported encryption types are SSLv1, SSLv2, SSLv3, and TLSv1.

Secure Sockets Layer (SSL) 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.

The nCipher nForce 300 card will be supported for the AIX, Solaris, and Windows platforms. Refer to product documentation for the complete description of all of the cards we will support when the product is generally available.

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 public key authentication part (happening in the initial SSL handshaking), but not for the DES encryption part used in normal data transmission afterwards. Most 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. The browser-to-WebSEAL SSL session is built whenever a user with a browser tries to communicate with 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 offloading of work from the WebSEAL host's processor to the card.

There is no Access Manager-specific documentation to cover this enhancement. The support of the card is simply a function of GSKit, so only the card's documentation is needed.

Other WebSEAL functionality

WebSEAL supports an e-community single sign-on functionality that enables Web users to perform a single sign-on and move seamlessly between WebSEAL servers in two separate secure domains. We take a closer look at e-community single sign-on in [Chapter 12, “Access control in a distributed environment” on page 311](#).

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 fail-over cookie feature is also supported by the Plug-In for Edge Server, which is discussed in 5.4.2, “Plug-In for Edge Server” on [page 153](#).

Architecture

The WebSEAL architecture is summarized in [Figure 5-5 on page 153](#). 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 the URL.

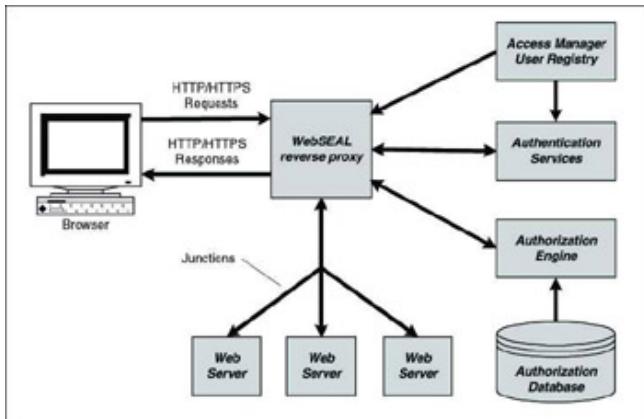


Figure 5-5: WebSEAL architecture

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.

5.4.2 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 fail-over cookies.

Architecture

[Figure 5-6](#) 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 5-5 on page 153](#)), the specific functionality and configuration of various components does differ.

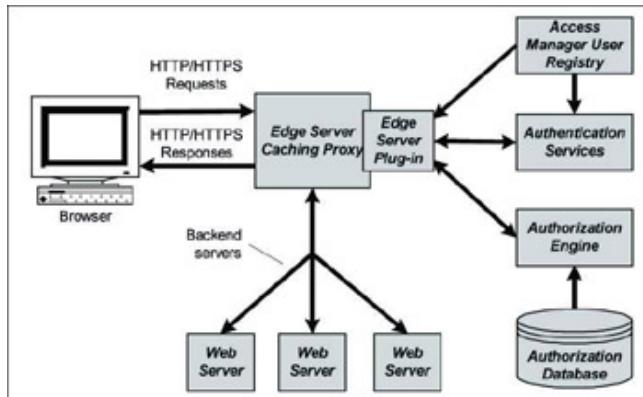


Figure 5-6: Plug-in for Edge Server architecture

5.4.3 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 5-2](#) summarizes the capabilities that are provided by this implementation based on a WebSEAL comparison.

Table 5-2: 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
Cross Domain Authentication Service (CDAS)	Supported
Interoperability with WebSEAL fail-over 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

Authentication support	
Authorization support	
ACL and POP policies	Supported
Tag/value support	Supported
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.

Authentication support

IIS on Windows 2000

- IIS 5.0
(need to clarify service packs and security fixes)
- Windows 2000 Server and Windows 2000 Advanced Server

Authentication support

IPlanet on Solaris	<ul style="list-style-type: none"> ▪ IPlanet 6.0 ▪ Solaris 7 (Sparc) ▪ Solaris 8 (Sparc) - “run at”
Apache	<ul style="list-style-type: none"> ▪ Apache 1.3.20 ▪ Red Hat Linux 7.1 (x86) ▪ Mod_ssl 2.8.4

Other

Directory support: IBM Directory Server, Netscape iPlanet Directory, MS Active Directory, and Domino NAB Registry	Supported
Virtual hosting	Provided by the host Web server
Install/configure/uninstall	Simple and intuitive
URL and HTTP protocol transparency	Supported

Authentication support

Globalization-languages supported

Same as Access Manager Base, except bi-directional languages will not be supported

[Figure 5-7 on page 157](#) shows an architectural overview of the Web server plug-in implementation.

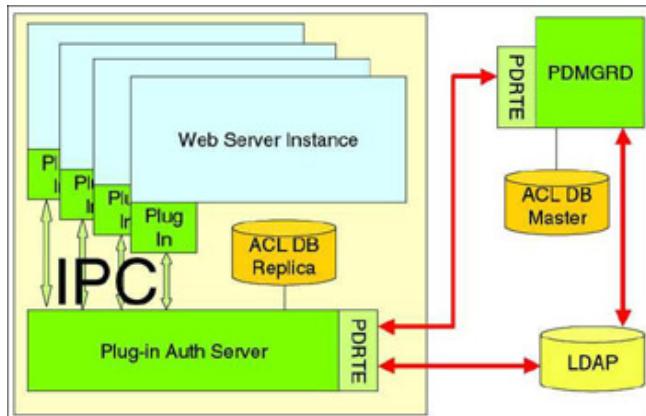


Figure 5-7: Access Manager Web server plug-in architecture

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 inter-process 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.

5.4.4 Other Resource Managers

There are several other Resource Managers available for Access Manager, some of which are targeted at a system-level integration and others at an application-level integration.

Access Manager for WebSphere Application Server

Access Manager provides a Java 2 Platform Enterprise Edition (J2EE) container-level integration with WebSphere Application Server 5.1.

Access Manager for WebSphere Application Server consists of the WebSphere 5.1 Authorization Plug-In and an associated migration utility. This J2EE integration provides an Access Manager-compatible implementation of the WebSphere Version 5.1 authorization table interface for most *workstation* platforms supported by WebSphere Advanced Edition. 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 [Chapter 11, “WebSphere application integration”](#) on [page 285](#).

Access Manager for BEA WebLogic Server

Tivoli Access Manager for WebLogic, Version 5.1, supports:

- BEA WebLogic Server version 7.0 SP2
- BEA WebLogic Server version 8.1 SP1

Tivoli Access Manager for WebLogic Version 5.1 provides a full security framework for BEA WebLogic Server using the Security Service Provider Interface (SSPI).

Note Tivoli Access Manager for WebLogic Version 5.1 does not support the BEA WebLogic Server *custom realm*. Support for the BEA WebLogic Server custom realm was part of Tivoli Access Manager for WebLogic Version 4.1.

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.

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 single sign-on 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 5.1*, SC32-1366.

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 5.1, a new entitlement service interface, has been added to the aznAPI that is called during the building of a credential. This entitlement service receives the basic user credential being created and is able to

specify a list of additional custom attributes to be added to the credential before it is returned to the application.

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 11.4, “Access Manager and WebSphere integration” on [page 294](#).

5.5.3 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.

5.5.4 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 5-8](#).

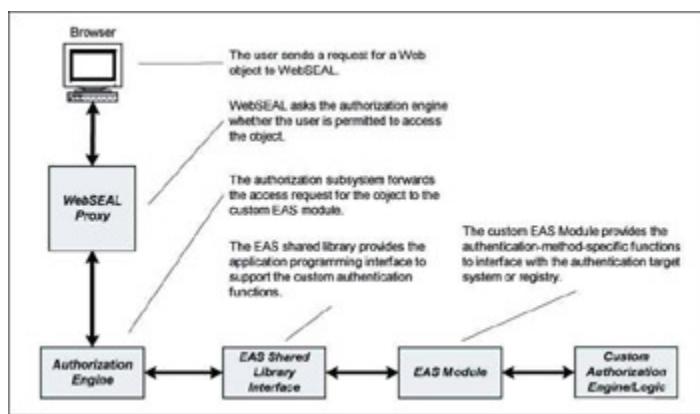


Figure 5-8: WebSEAL EAS architecture

5.5.5 Cross Domain Authentication Service

The Cross Domain Authentication Service (CDAS) is specific to WebSEAL. It provides a shared library mechanism that enables you to substitute the default WebSEAL authentication mechanisms with a custom process that ultimately returns an Access Manager identity to WebSEAL.

When WebSEAL determines that it must authenticate a user, an installation-specific CDAS can be invoked that performs the authentication using whatever mechanism is desired (for example, the user could be authenticated against a “foreign” user registry). Upon authenticating the user, the CDAS then “maps” the user to an identity defined in the Access Manager user registry (for example, one could log on

via a foreign registry as *joe* and then be mapped by a CDAS to the Access Manager user *fred*).

[Figure 5-9](#) shows an overview of the CDAS architecture.

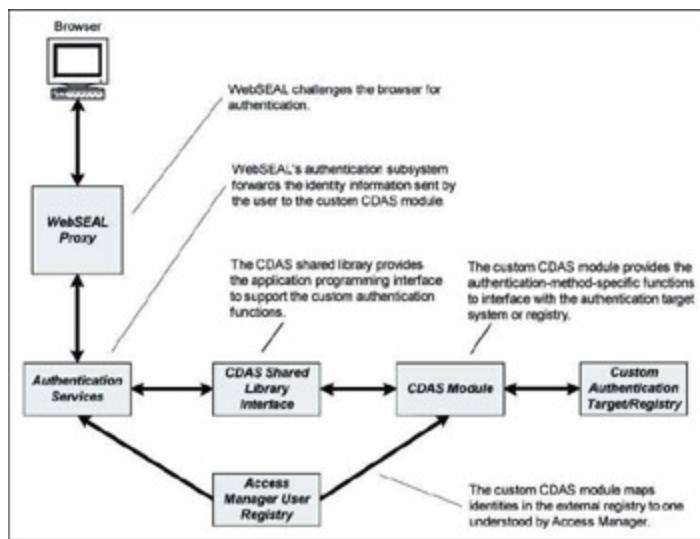


Figure 5-9: CDAS architecture

5.5.6 Cross Domain Mapping Framework

The Cross Domain Mapping Framework (CDMF) is a programming interface that may be used in conjunction with WebSEAL e-community SSO and Cross Domain Single Sign-On (CDSSO). It enables a developer to customize the mapping of user identities and the handling of user attributes when e-community SSO functions are used.

Conceptually, the mapping in a CDMF function works in a manner similar to a CDAS, 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.

Chapter 6: Access Manager Web-based Architecture

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 3.3.1, “Localizing a global vision with MASS” on [page 57](#) and 3.3.2, “Network zones” on [page 60](#).

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.

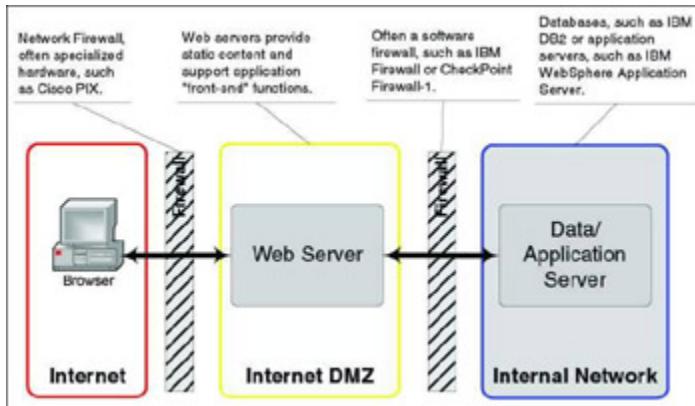


Figure 6-1: Typical advanced Web application architecture

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 IBM MASS approach.

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 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 MASS 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 Basic WebSEAL component interactions

As discussed in [Chapter 5, “Introduction to Access Manager components”](#) on [page 127](#), 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 6-2](#).

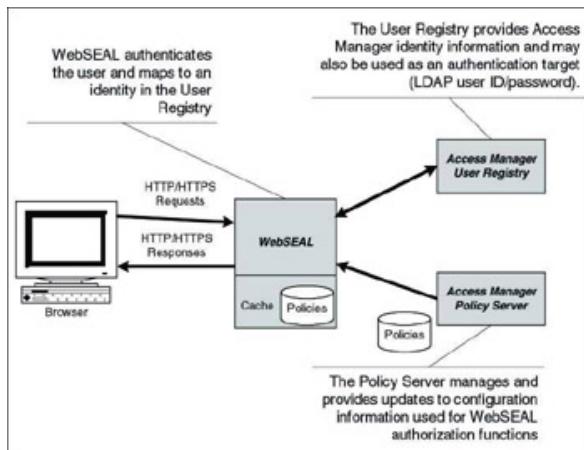


Figure 6-2: WebSEAL interaction with other Access Manager components

In the most basic of WebSEAL architectures, as shown in [Figure 6-3](#) on [page 171](#), 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.)

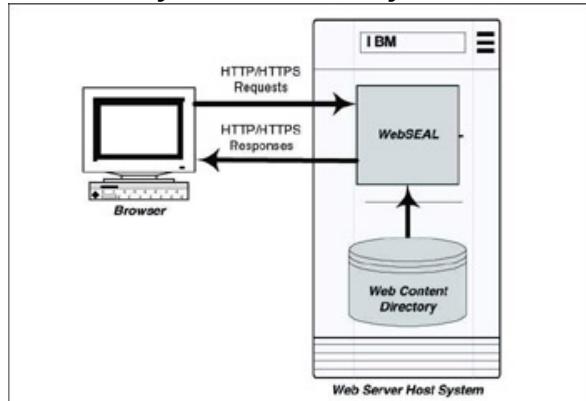


Figure 6-3: Direct serving of Web content from WebSEAL

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 6-4 on page 172](#) shows the basic flow involved in processing such a request.

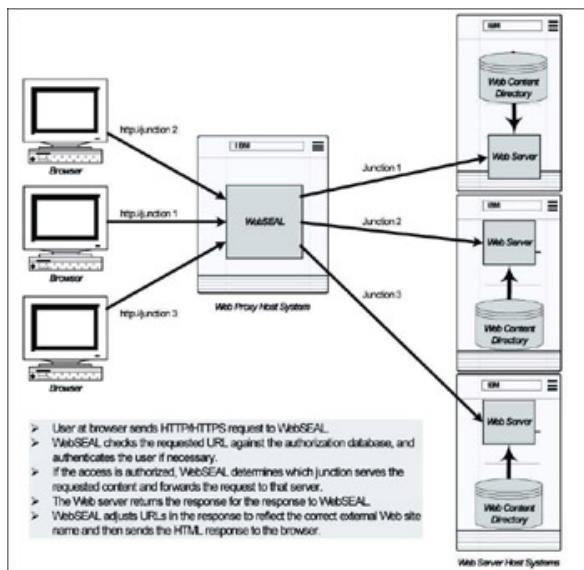


Figure 6-4: Basic WebSEAL proxy functionality

The flow in [Figure 6-4](#) 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 166](#), 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.

6.5 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 single sign-on 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 6-5](#) on [page 174](#).

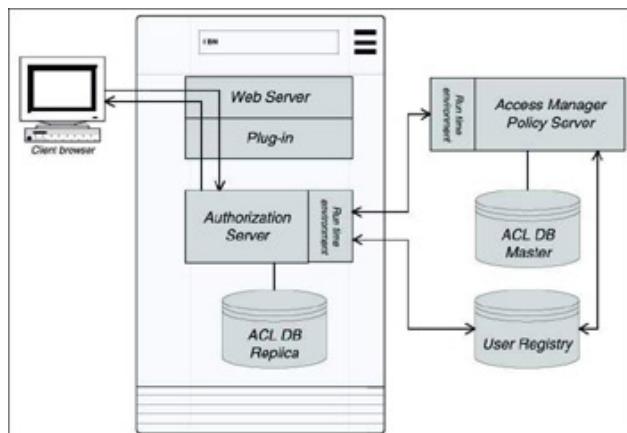


Figure 6-5: Basic Web Server Plug-in components

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 6-6](#). 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.

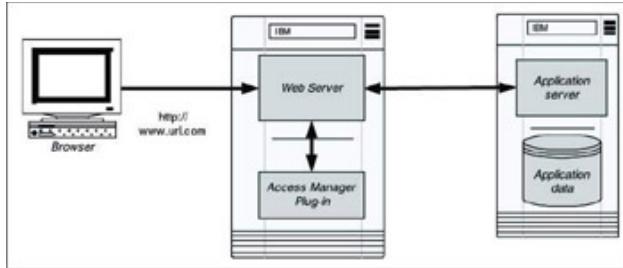


Figure 6-6: Plug-in logical architecture

Proxy Policy Server

The Proxy Policy Server enables Access Manager applications and authorization servers to connect to a Proxy Policy Server rather than the Policy Server. The addition of a Proxy Policy Server enables an architecture to be created where the only incoming SSL sessions to the Policy Server come from physical Proxy Policy Servers. This facilitates increased security because a firewall protecting the Policy Server only has to allow inbound connections from the Proxy Policy Server(s) rather than from all Tivoli Access Manager applications or authorization servers. The SSL session from Access Manager applications to the Proxy Policy Server(s) is independent from the SSL session from the Proxy Policy Server to the Policy Server.

[Figure 6-7](#) shows the connections (and the direction of flow) between the Policy Server, a Proxy Policy Server and an Access Manager application or authorization server.

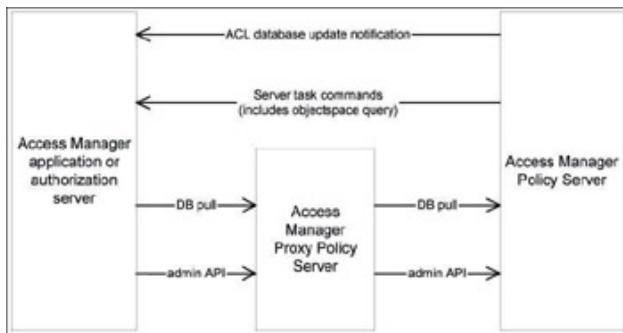


Figure 6-7: Communication flows using the Proxy Policy Server

All requests inbound to the real Policy Server go via the Proxy Policy Server. 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 Proxy Policy 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 Proxy Policy Server one time.

The ACL databases are cached in memory for security. There is no ACL policy database stored on the disk of the Proxy Policy Server that could be read (or modified) if the Proxy Policy Server were compromised.

The currency of the ACL database in the Proxy Policy 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 Proxy Policy 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.

6.6 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.

6.6.1 Network zones

In [Chapter 3, “IT infrastructure topologies and components” on page 49](#), 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 6-8](#) summarizes the general Access Manager component type relationships to the network zones discussed above.

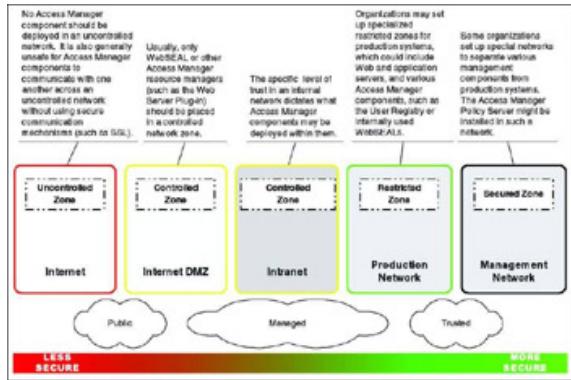


Figure 6-8: Network zones

6.6.2 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.

6.6.3 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 6-9](#) summarizes the guidelines for

placement.

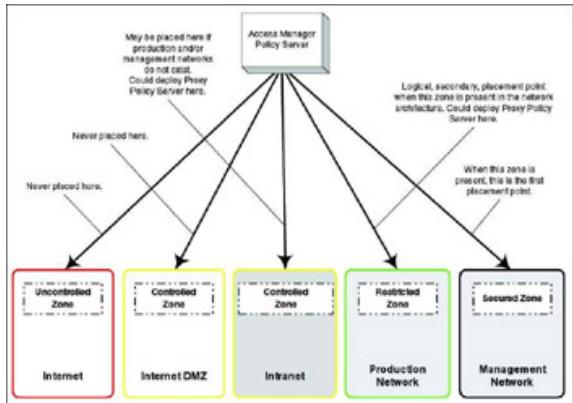


Figure 6-9: Policy Server placement guidelines

In the case of using the Proxy Policy Server it should be placed in a more trusted zone adjacent to the location of the Access Manager applications. For example, in [Figure 6-9](#) 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 Proxy Policy Server within the intranet so that no direct connections are allowed from the Internet DMZ to the management network.

Note Using the Proxy Policy 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 Proxy Policy 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 6-10 on page 182](#) summarizes network zone placement guidelines for the User Registry.

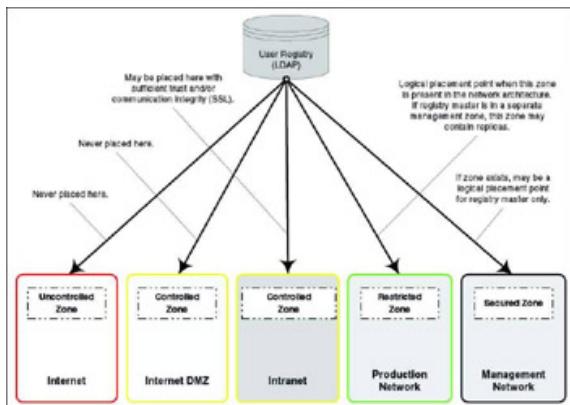


Figure 6-10: User Registry placement guidelines

[Figure 6-11 on page 183](#) shows an example of User Registry placement using network filtering rules to limit access.

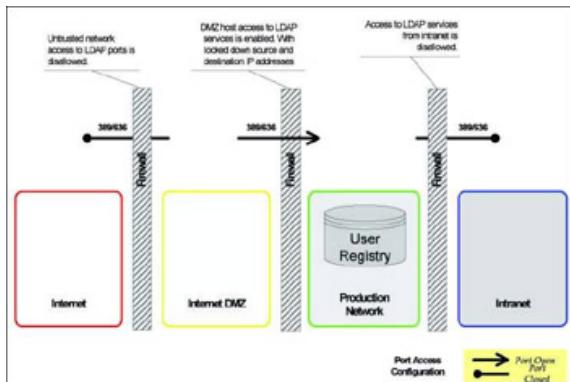


Figure 6-11: Restricting network access to User Registry

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 6-12 on page 184](#) 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.

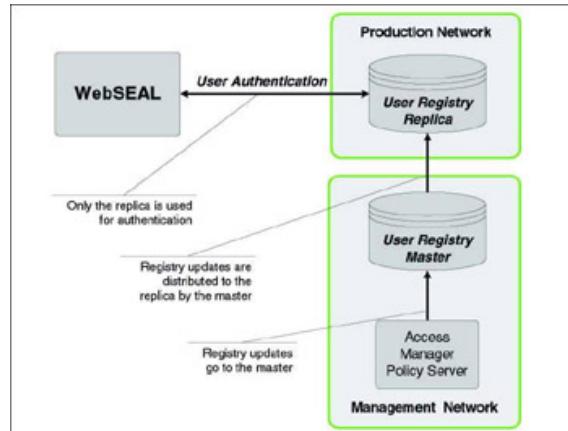


Figure 6-12: Separating User Registry read and write functions

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 6-13 on page 185](#) summarizes placement guidelines for the Web Portal Manager.

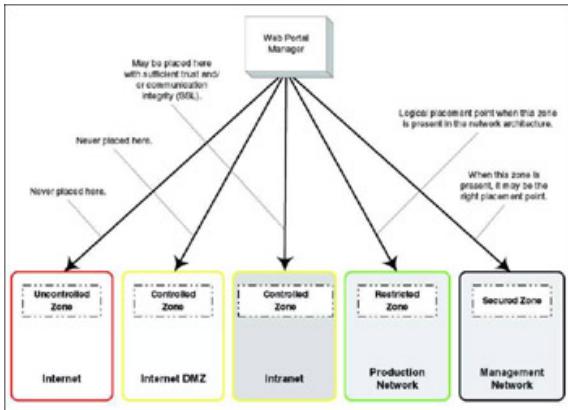


Figure 6-13: Web Portal Manager placement guidelines

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 6-14](#).

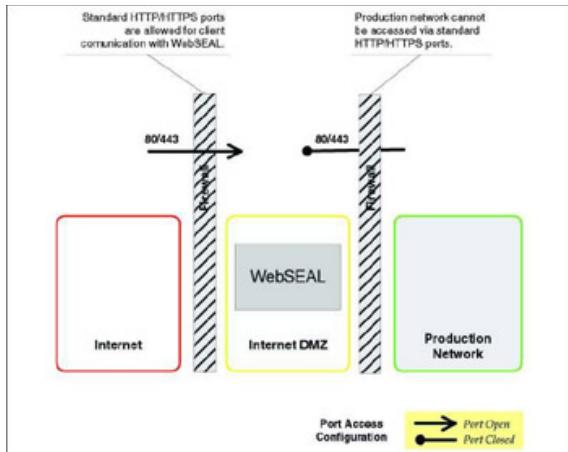


Figure 6-14: Restricting HTTP/HTTPS network traffic paths

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 6-15](#) on [page 188](#) summarizes the guidelines for WebSEAL placement.

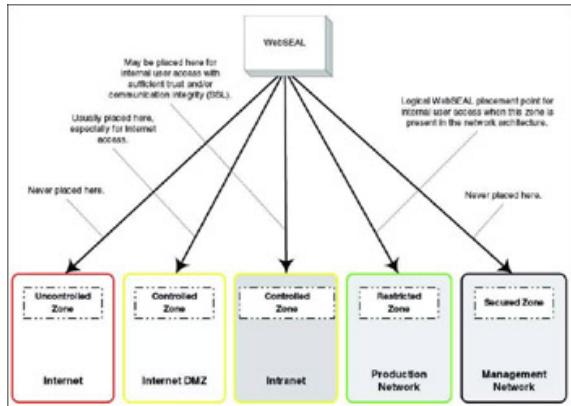


Figure 6-15: WebSEAL placement guidelines

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 6-16](#) on [page 189](#) summarizes the zone placement guidelines for Web servers that are junctioned via WebSEAL.

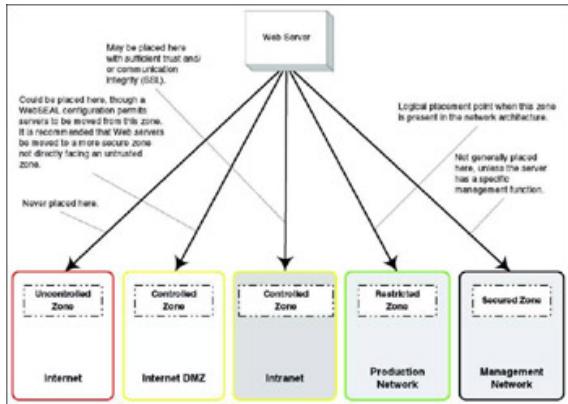


Figure 6-16: Web server placement guidelines

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 6-17](#) on [page 190](#).

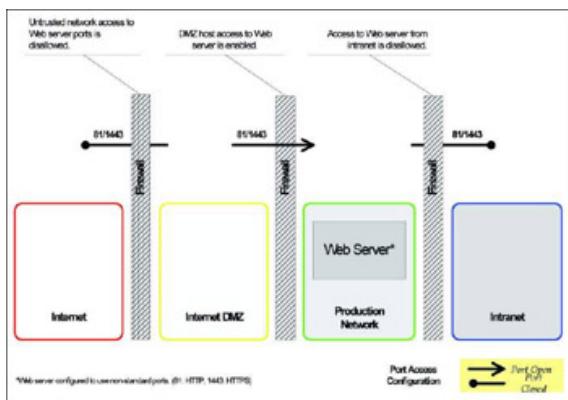


Figure 6-17: Limiting network access to Web servers

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 6-18 on page 191](#) 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.

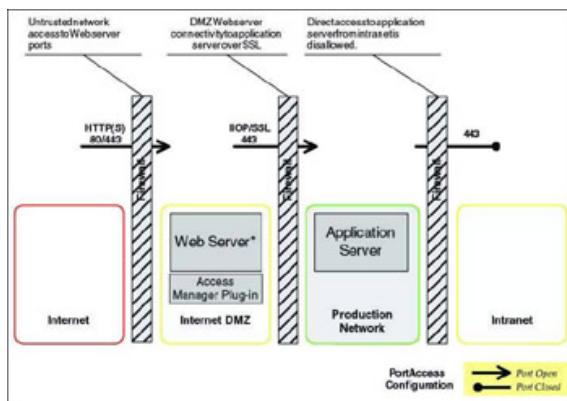


Figure 6-18: Plug-in architecture

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 6-19](#).

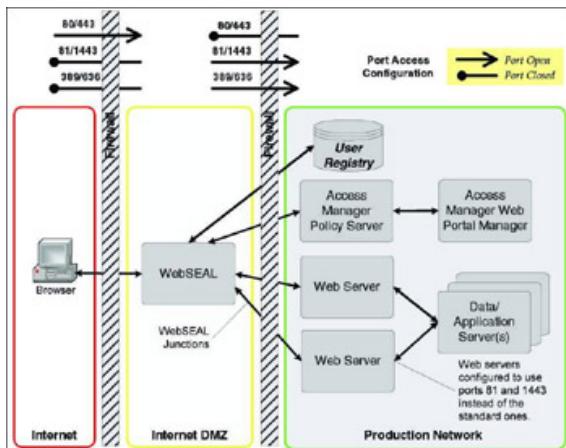


Figure 6-19: A sample Access Manager WebSEAL architecture

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.

6.6.4 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.

6.7 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.

6.7.1 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 6-20](#) on [page 195](#) shows a physical system layout mapping of the example architecture shown previously in [Figure 6-19](#) on [page 192](#). 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.

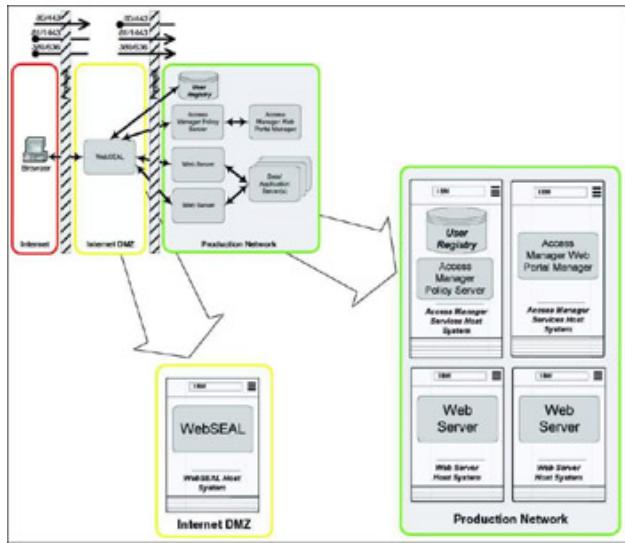


Figure 6-20: A sample physical component layout

6.7.2 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

Releases of Policy Director before Version 3.8 depended on time synchronization among the components (due to Distributed Computing Environment, or DCE). Though Access Manager no longer requires that time be synchronized, 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.

6.7.3 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.

6.7.4 Tivoli Access Manager port matrix

If you have worked with Tivoli Access Manager for e-business, you are familiar with the ports (80, 443) that are used for HTTP and HTTPS communication, respectively, between browsers and WebSEAL. The table in [Figure 6-21 on page 199](#) gives more detail about the use of ports for communication among Tivoli Access Manager for e-business components. The port numbers are defaults and (unless otherwise indicated) can be changed.

From A to B	B-pdrte	B-padmin	B-wsadmin	B-sdtp	B-pdpns	B-pdrte ^a	B-pdpns ^a	B-pdrte	B-pdpns	B-Edge IP	B-Edge PS	LDAP ^b
A-pdrte	7027 ^c		7224 ^c	7227 ^c	Receiving Port	Receiving Port	Receiving Port	Receiving Port	Receiving Port	7040	7099 or 636	
A-padmin	7131 ^c				Receiving Port	Receiving Port	Receiving Port	Receiving Port	Receiving Port	389 or 636		
A-wsadmin	7135 ^c									589 or 636		
A-analog	7139 ^c									389 or 636		
A-dpdns	7131 ^c											
A-pdpns	7134 ^c	7136 ^c										
A-pdpns	7138 ^c	7136 ^c										
A-edge	7135	7136			IPC	IPC	IPC					
A-pdrte	7135	7136			IPC	IPC	IPC					
A-Edge PT	7135									389 or 636		
A-LDAP	Receiving Port	Receiving Port	Receiving Port	Receiving Port				Receiving Port	Receiving Port			

Figure 6-21: Access Manager port matrix

Table footnotes:

1. AMWAS/PDWAS uses the remote Java client, so AMWAS is not listening on any ports.
2. This represents WebSphere Portal as a remote-mode JAAS application.
3. This entry refers to the case where there is a machine with just pd rte and the pd admin CLI on it (not the more general case of any server using pd rte as its foundation).
4. 389 for ldap; 636 for ldap-s. Note that if the LDAP server co-resides on the same box as Active Directory, then the LDAP ports must change, as they are in conflict with Active Directory and Active Directory's ports cannot be modified.

5. ACL DB replication notification and server task functions.
6. Incremented by 1 for each new WebSEAL instance. So, the second WebSEAL instance defined would listen on port 7235, and so on.
7. ACL DB replication.
8. To be 100% technically accurate, WebSEAL and the Web Server Plug-In communicate with the Policy Server via the AM Run Time Environment.
9. Only while the migration tool or pdwascfg is being run.
10. This is needed for pdadmin API to work on the WPS machine.
11. For Java pdpermission and pdprincipal calls. Also may be needed during registration of remote mode Java application.
12. To be 100% technically accurate, PDWPM, PDWAS, and PDWPS communicate with the Policy Server and/or the Authorization Server via the AM Java Run Time Environment.

A few points about the color-coding. A green cell with *Receiving Port* within indicates that the receiving component listens on a port designated in the original TCP/IP communication from the sender. A yellow cell with */PC* within indicates that the communication does not involve ports, but rather runs as an interprocess communication, in memory.

Here is a mapping from the abbreviated component names used in the table, to more official component names:

pdmgrd	Access Manager for e-business Policy Server
pdaclid	Access Manager for e-business Authorization Server
websealid	Access Manager for e-business WebSEAL Server

webpi	Access Manager for e-business Web Server Plug-In
pdwpm	Access Manager for e-business Web Portal Manager (Web-based management GUI)
pdwas	Access Manager for e-business WebSphere Application Server Component
pdwps	WebSphere Portal as an Access Manager for e-business application
pdjrte	Access Manager for e-business Java Runtime Environment
pdrte	Access Manager for e-business C Language Runtime Environment
Edge PI	Access Manager for e-business Edge Server Plug-in
<u>LDAP</u>	Access Manager for e-business LDAP Registry (by default, IBM Directory Server)

6.8 Access Manager in an 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 contingencies. 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 Risk Manager, which are discussed in [Part 3, “Managing identities and credentials”](#) on [page 417](#) and [Part 4, “Managing a security audit”](#) on [page 521](#), provide functionality complementary to Access Manager and can be of substantial value as components of an overall security solution.

Chapter 7: A Basic WebSEAL Scenario

Overview

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 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/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.

Attention As of April 2002, our fictitious domain name Stocks-4u.com was not reserved by anyone.

7.1.1 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 wish to deploy adequate capacity to address their anticipated loading over six months, and then incrementally add more as needed.

7.1.2 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

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.

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](#) on [page 204](#) shows the national Stocks-4u.com network.

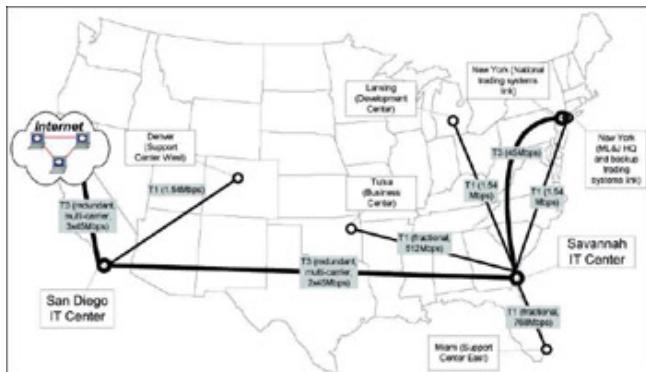


Figure 7-1: Stocks-4u.com data network

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.

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.

7.1.3 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.

7.1.4 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 wish 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 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.

7.1.5 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 that:

- 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.

From this, we can easily address an initial WebSEAL-based Access Manager architecture for Stocks-4u.com.

7.1.6 Access control architecture

As we know it today, the diagram in [Figure 7-2 on page 208](#) summarizes the existing security architecture deployed by Stocks-4u.com with multiple Web server host systems deployed in the Internet DMZ.

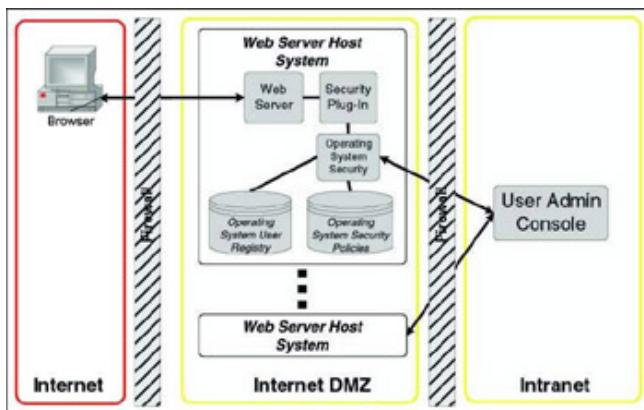


Figure 7-2: Current Stocks-4u.com architecture

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.

Initial architecture approach

Recalling the discussions in [Chapter 5, “Introduction to Access Manager components”](#) on [page 127](#) and [Chapter 6, “Access Manager Web-based architecture”](#) on [page 165](#), we know that we will place a WebSEAL server in the DMZ, which will provide for Internet user access. We also know that the user registry, policy server, and Web Portal Manager (WPM) should not reside in the DMZ, and we will place those components in the San Diego center internal production network.

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 a good thing, as 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.

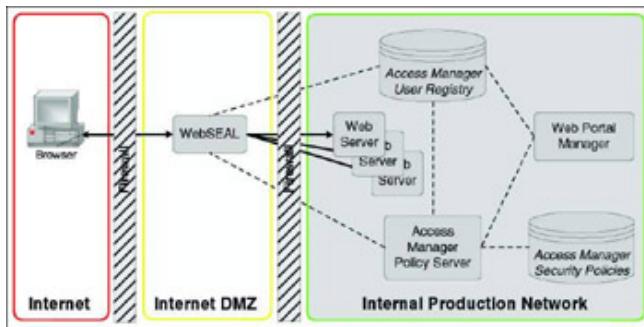


Figure 7-3: Initial WebSEAL architecture

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.

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 will 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.

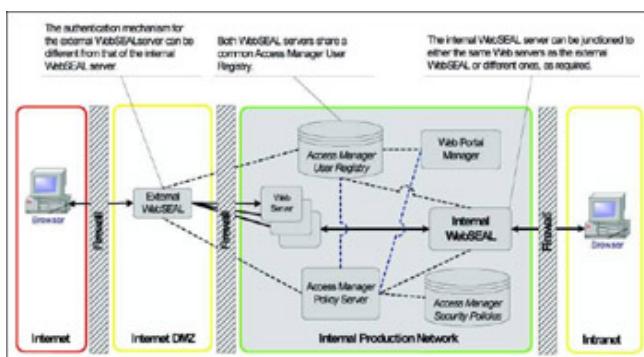


Figure 7-4: WebSEAL security architecture with internal WebSEAL

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 Web-based architecture”](#) on [page 165](#). 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.

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 wish 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 have addressed the communication among the components, our new architecture is shown in [Figure 7-5](#).

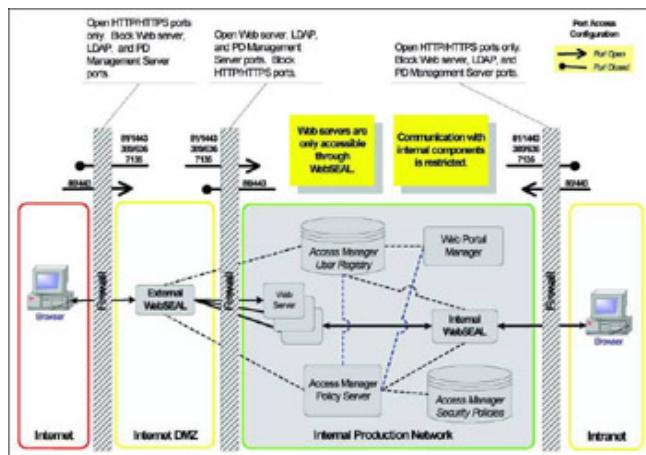


Figure 7-5: Detailed WebSEAL security architecture with internal WebSEAL

7.1.7 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.

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.

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 NT® or Windows 2000 platform as well as on AIX and Solaris. 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 on page 214](#) shows the final physical architecture for the initial deployment.

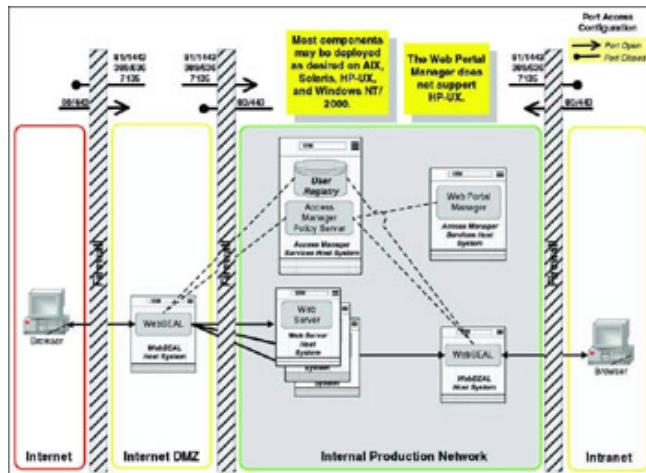


Figure 7-6: WebSEAL physical architecture

7.1.8 Architectural summary

In this chapter, we have 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.

Chapter 8: Increasing Availability and Scalability

Overview

In this chapter we continue the discussion from the previous section 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 portal (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 WebSEAL portals, 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 fail-over 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 218](#), 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.

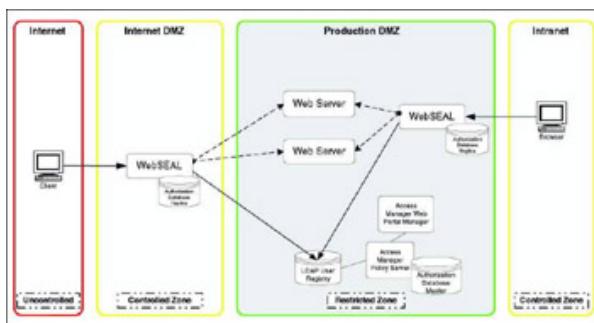


Figure 8-1: Initial Web architecture

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, at least it does not affect the availability of your Web site. WebSEAL can still perform all necessary authorization operations because it uses 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 NT or Windows 2000 to other powerful platforms such as Solaris, HP-UX, AIX, 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 on page 220](#), we describe the different areas with their solutions.

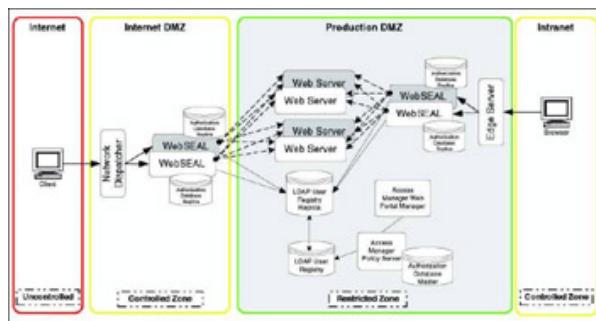


Figure 8-2: Server replication to increase availability

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 on page 221](#), there are several places where the configuration must be duplicated.

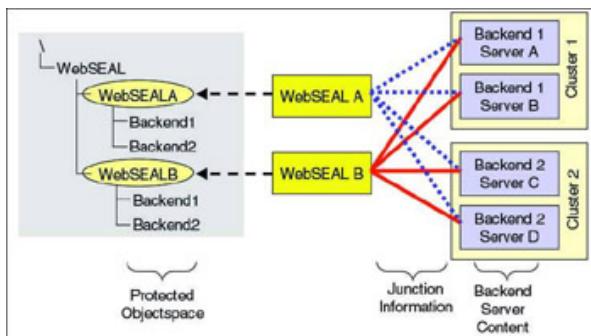


Figure 8-3: WebSEAL availability overview

- 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 223](#).

- 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, as shown in [Figure 8-4 on page 222](#).

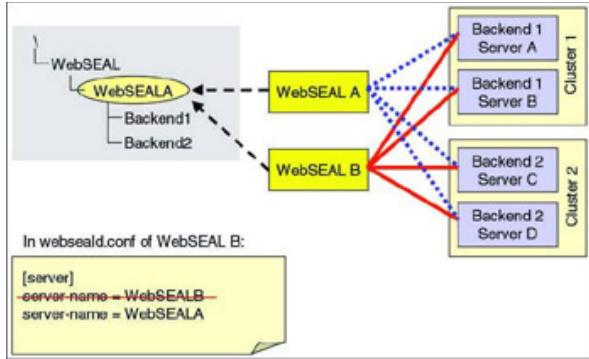


Figure 8-4: WebSEAL availability configuration

Configuring a WebSEAL cluster

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 /index.html, the object checked is /WebSEAL/WEBSEALB/index.html. However, if the server-name parameter in webseald.conf is changed to WEBSEALA, WEBSEALB will now check the object /WebSEAL/WEBSEALA/index.html.

The portion of the object space under /WebSEAL/WebSEALB is now redundant; all checks are done against the objects under /WebSEAL/WEBSEALA. As long as the file space of both servers is identical (which means that they have the same junctions and the same back-end servers), then this will be fine and will remove the need to duplicate work. 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.

After renaming the server name value WebSEALB to WebSEALA in the webseald.conf file on WebSEALB, the server object in the name space is now meaningless and can be removed:

```
pdadmin>object delete /WebSEAL/WebSEALB
```

WebSEAL fail-over cookies

Fail-over 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 fail-over cookies over HTTP or HTTPS.

The failover-cookies-keyfile entry in the webseald.conf file points to a file containing a triple DES key created using the cdsso_key_gen utility. This keyfile must be shared by all WebSEAL servers in the Access Manager secure domain that the user might be redirected to in the event of a failure.

Note The processing of fail-over cookies is processor-intensive and should only be used for failure recovery. They should not be used for load balancing.

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 5.1*, SC32-1359.

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 221](#). 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.

Authorization Server availability

Although not initially depicted in the basic scenario in [Figure 8-2](#) on [page 220](#), assume for now that we have 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 8-5.

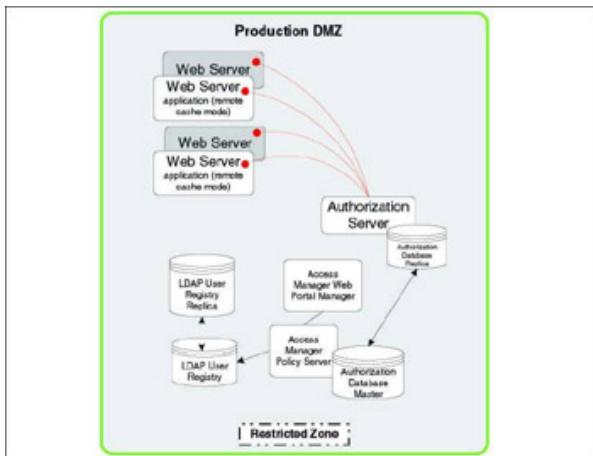


Figure 8-5: Authorization Server scenario for Stocks-4U.com

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 8-6](#) on [page 225](#).

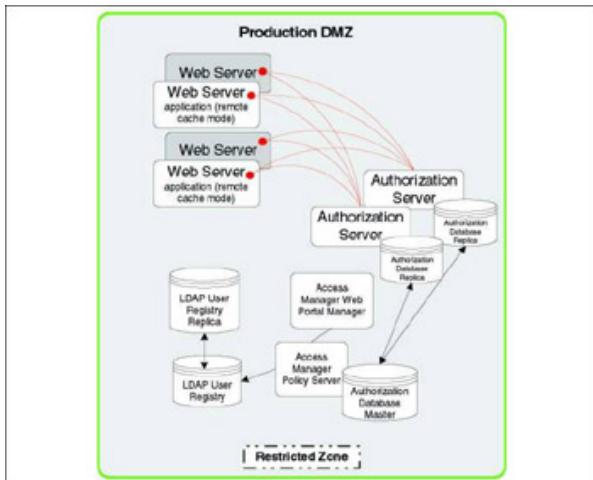


Figure 8-6: Authorization Server scenario with high availability

After implementing a second Authorization Server, you would only need to configure your aznAPI applications to be aware of the new replica. This is done by executing the **bassslcfg - add_replicas** command.

Another way of implementing this particular scenario could be by configuring the applications to run in local cache mode, shown in [Figure 8-7 on page 226](#). By doing this, the aznAPI calls would not go out to a remote Authorization Server for access control checks, but instead use the local authorization database replica.

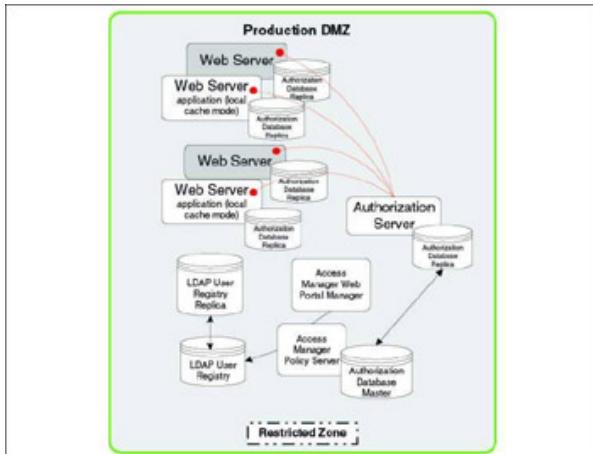


Figure 8-7: Authorization Server scenario on local cache mode

User registry availability

The IBM Directory Server supports the concept of master and replica LDAP servers. This is discussed in more detail in 4.3, “IBM Tivoli Directory Server” on [page 104](#).

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 configuration for multiple LDAP directories

Access Manager connects to the LDAP master server when it starts up. If the LDAP master server is down for any reason, 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 fails over to a replica server when it cannot connect to the master server.

You can find the configuration parameters for LDAP failover in the [ldap] stanza of the ldap.conf configuration file:

UNIX	/opt/PolicyDirector/etc/ldap.conf
Windows	install-path\etc\ldap.conf

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 Directory Server (LDAP) supports the existence of a single read-write master LDAP server. iPlanet Directory Server supports multiple read-write LDAP servers. Access Manager treats the iPlanet supplier server as the master server for configuration purposes. The active configuration lines in the ldap.conf file represent the parameters and values for this

master LDAP server. You determine these values during Access Manager configuration. For example:

```
[ldap ]
enabled =yes
host =outback
port =389
ssl-port =636
max-search-size =2048
```

If you make a change to the LDAP database, such as adding a new user account through the WPM, Access Manager always uses the read-write (master) LDAP server.

2. Replica server configuration

IBM Directory Server (LDAP) supports the existence of one or more read-only replica LDAP servers. iPlanet Directory Server (LDAP) supports the existence of one or more read-only replica LDAP servers referred to as consumers.

You must add lines to the [ldap] stanza that identify any replica servers available to Access Manager. Use the following syntax for each replica:

```
replica = ldap-server, port, type, preference
```

Changes to the *ldap.conf* file do not take effect until you restart Access Manager. More about configuration can be found in the *IBM Tivoli Access Manager for e-Business Version 3.9 Base Administration Guide*, GC23-4684.

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 fail-over 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.

In Access Manager 5.1, a new configuration option is added for configuring additional Policy Servers to act as standby Policy Servers.

No tools are supplied to manage the failover or fallback—this function simply provides a supported way to configure additional Policy Servers into an Access Manager environment.

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 8-8 on page 229](#) 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.

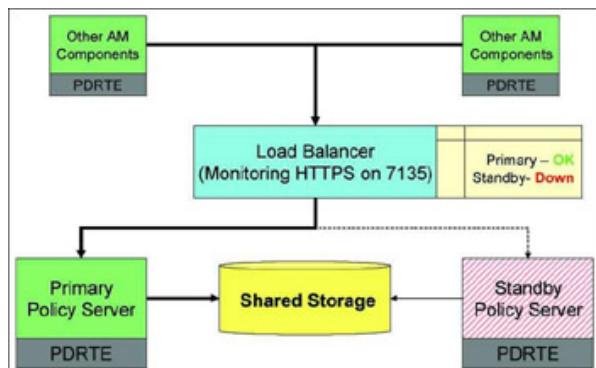


Figure 8-8: Standby policy server configuration using a load balancer Other AM Components

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 and POPs). 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, more than one Web Portal Manager should be deployed. This can be the case if you have delegated administration for your business partners to external domain administrators or if you are in the ASP business.

WPM runs on a WebSphere Application Server base, so you should deploy the application using another WebSphere Edge Server dispatcher unit in front of your multiple WPM or set up a WebSEAL junction for the WPM application in order to use the Edge Server/WebSEAL deployment as a frontage for high-availability access.

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 component 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.

WebSEAL scalability

To add another WebSEAL machine to your existing cluster:

- Install and configure a new WebSEAL server; an initial copy of the authorization database gets copied from the Policy Server.
- Edit the [server] stanza in the webseald.conf file, as shown in [Figure 8-4 on page 222](#).
- Copy the existing junction definitions (XML files) to the new server.

- Add the new WebSEAL IP address to the load balancing table of your IBM Network Dispatcher or Cisco Local Director.
- Install and configure the necessary certificate information if you are using SSL communication, mutual authentication with your back-end Web servers, or fail-over cookies.
- Start the new WebSEAL.

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.

Authorization Server scalability

To add another Authorization Server component to your infrastructure:

1. Install a new Authorization Server; an initial copy of the authorization database gets copied from the Policy Server.
2. Define this server as a new Authorization Server replica to your applications by using the **bassslcfg - add_replicas** command.
3. Install and configure the necessary certificate information if you are using SSL communication.

The new Authorization Server will immediately be available to receive authorization requests from your applications. This way, you can easily extend your application infrastructure.

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 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.

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 Base Administration Guide Version 5.1*, SC32-1360.

For further capacity and availability discussion, refer to the *IBM Tivoli Access Manager for e-business Problem Determination Guide Version 5.1*, SC32-1352, and the *IBM Tivoli Access Manager for e-business Performance Tuning Guide Version 5.1*, SC32-1351.

Chapter 9: Authentication and Delegation with Access Manager

Overview

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 groundwork for the scenario in [Chapter 11, “WebSphere application integration” on page 285](#).

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 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 will 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.
- It is recommended 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.

Note You can perfectly integrate this and other

intrusion detection tools with the IBM Tivoli Risk Manager centralized auditing infrastructure. To learn more about this solution refer to Part4, “[Managing a security audit](#)” on [page 521](#).

We have already focused on general WebSEAL architecture issues in 5.4.1, “WebSEAL” on [page 148](#), as well as throughout [Chapter 6, “Access Manager Web-based architecture”](#) on [page 165](#), and [Chapter 7, “A basic WebSEAL scenario”](#) on [page 201](#). In this chapter, we concentrate on the different authentication and delegation mechanisms that can be utilized with WebSEAL.

9.1 Typical business requirements

In addition to the typical business requirements that were described in 6.2.1, “Typical business requirements” on [page 167](#), 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 168](#), which were driven by an overall Web security approach, we want to add the following concerns from the authentication aspect.

Here are some of the technical requirements for authentication that WebSEAL has to 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 identify 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.

- Delegation

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. This aspect of forwarding a user's credentials is also referred to as *single sign-on*.

- 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 best way to achieve the design objectives is by using a reverse Web proxy with sufficient security functions in front of the existing Web application servers. [Figure 9-1 on page 239](#) shows a basic architecture for protecting Web applications.

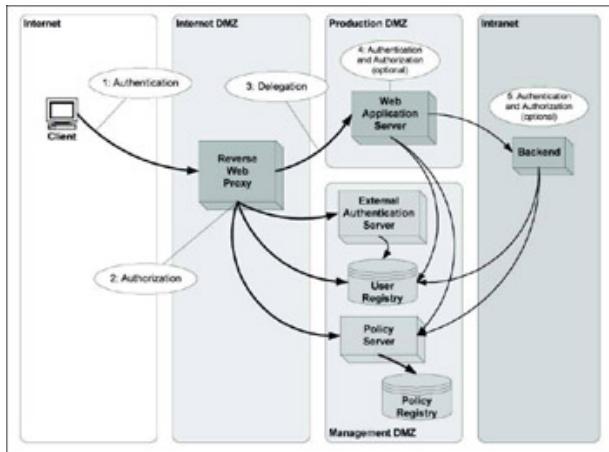


Figure 9-1: Reverse proxy flow for authentication, delegation, and authorization

The reverse proxy is used as a mediator between the end 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).
- 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 215](#).

9.3.1 Authentication and delegation mechanisms

This section presents the basic principles of authentication and delegation mechanisms that are used by WebSEAL 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. Delegation is the process of forwarding information about a user's identity in a secure way to another system. WebSEAL can enforce certain types of user authentication and can use several delegation mechanisms to forward user requests together with user information to a Web application server.

[Figure 9-2 on page 241](#) gives an overview of the various authentication and delegation mechanisms supported by WebSEAL. It depicts the available authentication schema between a user and WebSEAL, as well as the authentication between WebSEAL and other back-end application servers. The different mechanisms are discussed in greater detail in 9.4, “Supported WebSEAL authentication mechanisms” on [page 246](#).

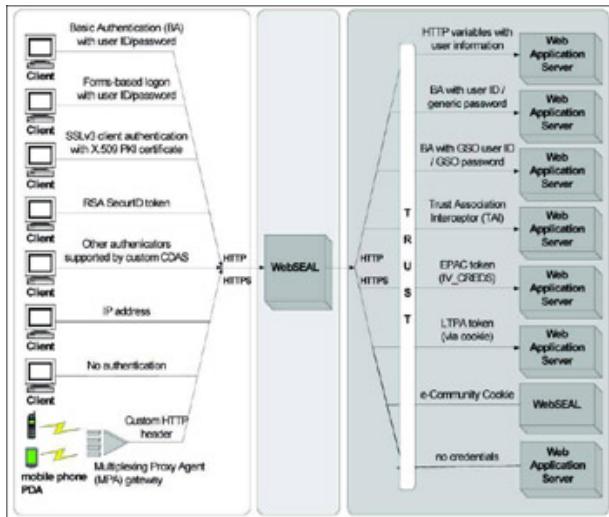


Figure 9-2: Access Manager authentication methods with WebSEAL

Look again at [Figure 9-1 on page 239](#) to follow the steps of the authentication process:

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.
5. Based on the forwarded user credentials, the back-end application server can proceed with further, more fine-grained authorization decisions.

WebSEAL provides enough flexibility to support multiple authentication and delegation mechanisms to act as a reverse Web proxy between different user groups and different types of Web application servers in a secure way.

The left portion in [Figure 9-2](#) lists all authentication mechanisms available between a user and WebSEAL. The right side lists all delegation or single sign-on mechanisms between WebSEAL and another Web application server.

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 any authentication mechanism with any delegation mechanisms.

9.3.2 Trust

A single WebSEAL server may be configured for three 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.

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 243](#) shows a list of Web server products that can be protected with Access Manager's WebSEAL using some of the mechanisms that we have listed.

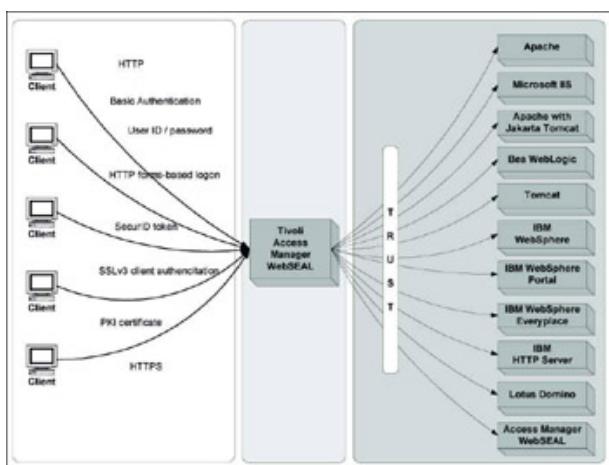


Figure 9-3: Overview of Web server products protected with WebSEAL

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 has also been addressed in [Chapter 6, “Access Manager Web-based architecture” on page 165](#).

9.3.3 Generic authentication mechanism with WebSEAL

Before going into the specific authentication model details, we use [Figure 9-4 on page 244](#) to look at a generic picture of the WebSEAL authentication model.

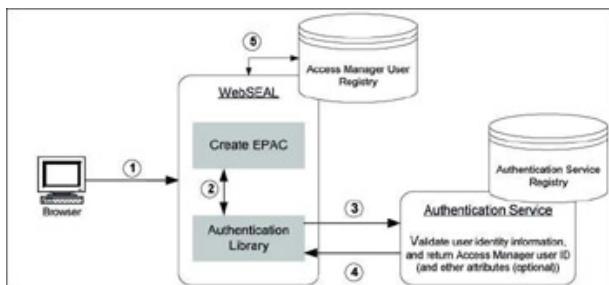


Figure 9-4: Generic WebSEAL authentication model

The following steps explain [Figure 9-4](#).

1. The user presents his identity information to WebSEAL.
2. WebSEAL 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. WebSEAL now uses the Access Manager user registry to create the Access Manager credential—Extended Privilege Attribute Certificate (EPAC)—that is cached for the duration of the session and used for any authorization decisions.

Note The EPAC is a token that details the user's identity and the roles that he can use. It can also contain additional attributes if required. EPACs are often misunderstood, are not encrypted or signed, and do not time out. They should only be trusted from within the Access Manager secure domain.

9.3.4 Generic delegation mechanism with WebSEAL

As discussed in 5.4.1, “WebSEAL” on [page 148](#), WebSEAL junctions provide powerful capabilities for managing access to multiple Web application servers through a common access portal. [Figure 9-3](#) on [page 243](#) shows the different junctions WebSEAL can establish.

If you want to delegate further authentication and/or 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 WebSEAL) 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, not for WebSEAL to portal.

Global 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.

WebSEAL 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 Supported WebSEAL authentication mechanisms

This section shows the authentication mechanisms that are supported by WebSEAL to protect access to a Web environment. All mechanisms in this section can be combined with any of the delegation mechanisms in the [next chapter](#) to make the connection between a user and a Web application.

WebSEAL uses the concept of Plug-on Authentication Modules (PAMs) to use different authentication methods. The programming interface is now available so users can write their own modules.

The following Plug-on Authentication Modules exist in Access Manager:

passwd-ldap	Password authentication (Forms/BasicAuth)
token-cdas	Token authentication (SecureID)
cert-ldap	SSL client certificate authentication
http-request	HTTP header authentication
cdsso	e-Community 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 WebSEAL 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. WebSEAL 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, an EPAC 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 will

automatically log on to WebSEAL 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, WebSEAL 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 on the send or logon button, the form is returned to WebSEAL using an HTTP POST request. WebSEAL 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, WebSEAL presents a customizable logout form to a user. After the user confirms the logout, the session is considered closed and the EPAC is deleted from the WebSEAL 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 WebSEAL configuration files.

9.4.3 Authentication with X.509 client certificates

In response to a certificate request from WebSEAL, 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 has selected a certificate, it is passed to WebSEAL, 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 WebSEAL, WebSEAL pulls the user information from the Access Manager user registry and builds the EPAC.

If you configure Access Manager to use X.509 client certificates for authentication, but the user does not have a certificate available,

WebSEAL can fall back to basic authentication, if required.

9.4.4 Authentication with RSA SecurID token

Access Manager includes a Cross Domain Authentication Service (CDAS) 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 WebSEAL, the client enters a user name and pass code. The pass code consists of:

- 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 token CDAS is complete. The CDAS does not perform identity mapping, but simply returns to WebSEAL 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

WebSEAL and the Web Server Plug-in support the SPNEGO 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 WebSEAL or the Web Server Plug-in using Kerberos authentication mechanisms.

Support for Kerberos authentication in WebSEAL and the Web Server Plug-in has been implemented specifically to support a Windows desktop single sign-on solution. This solution requires that the WebSEAL or the Web Server Plug-in server be configured into 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 WebSEAL or the Web Server Plug-in.

This means that a user accessing WebSEAL or the Web Server Plug-in with an IE browser can achieve single sign-on to Access Manager Web resources based on the authentication he provided to log in to the Windows domain on their local machine.

Kerberos authentication, which uses Active Directory services, is supported by WebSEAL running on Windows, AIX, or Solaris. It is also supported by the Web Server Plug-in running on Windows, AIX, Solaris, Intel Linux, and Linux for zSeries.

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 Custom authentication using CDAS

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 WebSEAL. 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.

WebSEAL provides a capability referred to as *Cross Domain Authentication Service (CDAS)* in order to meet these requirements.

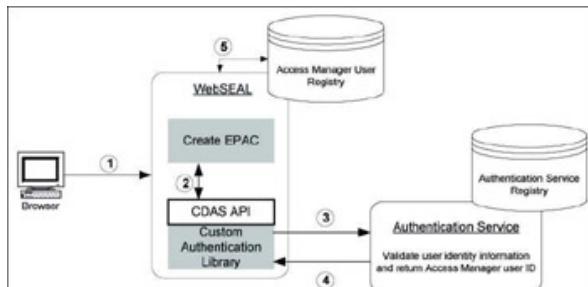


Figure 9-5: WebSEAL authentication model with CDAS

As shown in [Figure 9-5 on page 249](#), the CDAS enables you to substitute the default built-in WebSEAL authentication mechanism with a highly flexible shared library mechanism that allows custom handling and processing of client authentication information. The CDAS application programming interface (API) is available for download by registered customers.

You can customize the CDAS shared library to handle authentication data according to your security requirements given the following options.

The custom CDAS 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 build a custom CDAS. This method enables you to authenticate clients who are not direct members of the Access Manager security domain. In that case, the custom CDAS 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 CDAS returns an Access Manager identity to WebSEAL for querying the Access Manager user registry and creating an EPAC.

9.4.7 Entitlement service interface

In Access Manager 5.1, a new entitlement service interface has been added to the aznAPI that is called during the building of a credential. This service replaces the current CDAS-based approach to adding information from the user registry to a users credential for tag-value support. This entitlement service receives the basic user credential being created and is able to 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 9-6 on page 251](#) 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 WebSEAL and the Web Server Plug-in.

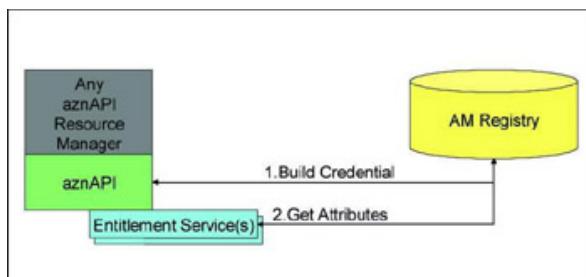


Figure 9-6: Entitlement service

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.

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.

9.4.8 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. WebSEAL can take this identity and create a credential for the user.

WebSEAL assumes that custom HTTP header data has been 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.

By default, this shared library is built to map data from trusted proxy headers.

9.4.9 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.

9.4.10 No authentication

Any user who can reach WebSEAL 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. WebSEAL can force unauthenticated users to use another authentication method when selecting certain protected URLs.

All users who can reach WebSEAL might already have enough permissions to contact certain junctioned Web servers. For example, if WebSEAL 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.

9.4.11 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 WebSEAL, the information across this channel initially appears as multiple requests from one client. WebSEAL must distinguish between the authentication of the MPA server over SSL and the additional authentication requests for each individual client.

Because WebSEAL maintains an SSL session state for the MPA, it cannot use SSL session IDs for each client simultaneously. WebSEAL instead authenticates clients using HTTP authentication techniques over SSL.

If the user is authenticated at the EWG, for example, to a RADIUS Server, then WebSEAL can be configured to receive an “authenticated ID” from the gateway and not re-authenticate the user.

WebSEAL has support for the Entrust Proxy and the Nokia WAP gateway.

9.5 WebSEAL delegation mechanisms

After a user has been authenticated by WebSEAL and an authorization decision has been made, WebSEAL has to forward the user's request to a back-end Web application server. If needed, WebSEAL 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 to use proprietary mechanisms when talking to specific server products. WebSEAL 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 WebSEAL 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, WebSEAL's realm is to provide the single sign-on functionalities for Web infrastructures. Acting as a Web reverse proxy to the company's Web environment, WebSEAL 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 WebSEAL-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 WebSEAL.

9.5.1 Tivoli Global Sign-On (GSO) lockbox

Most Web applications support basic authentication 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 201](#), 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, legacy 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.

The integration is achieved by creating SSO-aware junctions between WebSEAL and Web servers hosting the applications. GSO resources and GSO resource groups must first be created in Access Manager for every application. When WebSEAL receives a request for a resource located on the SSO-junctioned server, WebSEAL queries the Access Manager user registry for the appropriate authentication information. The user registry contains a database of 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, the following issues should be considered:

- Synchronization of the user IDs and passwords in the application's home user registry and Access Manager user registry.
- Storage of SSO passwords in the Access Manager user registry in clear, as they should be passed through to the application in clear. (They could be protected from the disallowed access, such as LDAP ACLs.)

A special situation emerges if Access Manager and the secured application share the same repository for storing user data, as shown in [Figure 9-7](#) on [page 256](#). An LDAP directory is the most suitable platform for maintaining application-specific information about users and groups. Given compatible LDAP schemas^[2], 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.

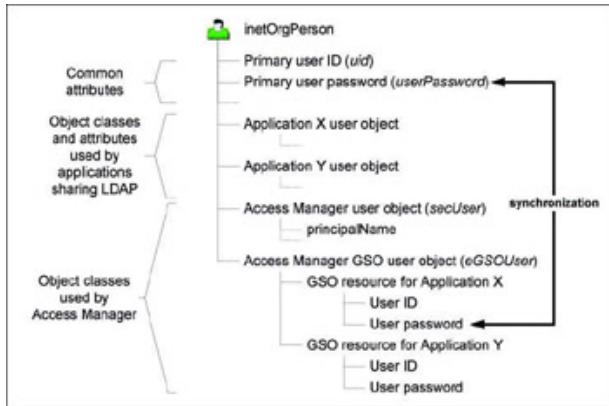


Figure 9-7: LDAP shared by Access Manager and other applications

The following issues should be considered while looking for solutions for integrating Access Manager and Web applications using the same LDAP repository:

- Main user passwords are allowed to be in clear. (Keep in mind, Access Manager GSO passwords are always in clear.) The possibility of protecting LDAP data based on ACLs always exists.
- Changing the main password should be reflected in the change of the GSO password for a particular user. This can happen immediately, for example after a user changes his password^[3], or in a batch run on a regular basis. The last situation presumes that main passwords are in clear.

Another way to resolve the LDAP “bind-issue” while sharing the same LDAP repository between Access Manager and secured Web applications is maintaining separate user entries. For example, a different subset of users is defined and maintained for Access Manager and its secured application. A user may have the DN=CN=Jon Doe,O=IBM,C=US and DN=CN=Jon Doe,OU=Access Manager,O=IBM,C=US for use by applications and Access Manager respectively, as shown in [Figure 9-8](#).

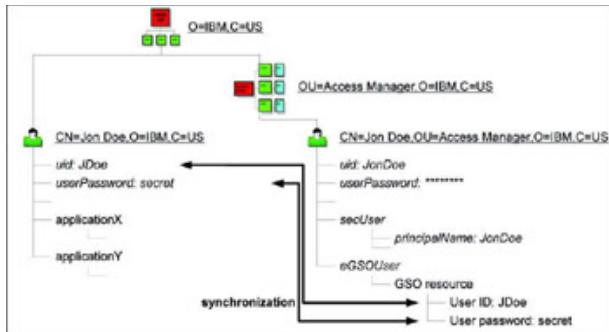


Figure 9-8: Shared LDAP with separate user entries

As a result, while performing authentication, the application will try to bind using its own user IDs and passwords. The GSO user IDs and passwords could be kept in sync more easily with those maintained by an application. The trade-offs of this solution are:

- The need to maintain the user information sets per application sharing the same LDAP.
- As the same user identity would exist multiple times, it would raise the direct cost if the licensing of the LDAP software is on a per-user basis.

9.5.2 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.

9.5.3 Junction without authentication

This may be useful if WebSEAL does all of the authentication and authorization and there is no need to forward any information to the back-end servers.

This scenario seems applicable either for servers without any reliable security functions or where there is no need of extra back-end authentication and authorization (for example, providing only static Web pages). Nevertheless, this approach requires full trust toward WebSEAL, and the back-end servers should be configured to accept only incoming requests from WebSEAL proxies.

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 Sign-On \(GSO\) lockbox” on page 254](#)) 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

WebSEAL 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

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 a WebSEAL reverse proxy can be understood by other WebSEALs in the same Access Manager security domain and even beyond (see [Chapter 12, “Access control in a distributed environment” on page 311](#)). However, the credential exchange with the junctioned Web server is not necessarily trivial mainly due to the lack of standardization. Kerberos, PKI, DCE, and Active Directory (with related products) are the most well-known security technologies, providing security interoperability for different platforms and applications, including Web-based environments; however, the applications have to be enabled for that. Not less important is the fact that these technologies do not interoperate seamlessly with each other; neither do the applications.

In order to support the interoperability of Web applications, WebSEAL today uses a generic HTTP-based interface as described in the previous sections.

9.5.6 Using LTPA authentication with WebSEAL

WebSEAL can provide authentication and authorization services and protection to an IBM WebSphere or Lotus Domino environment. When WebSEAL is positioned as a protective front end to WebSphere or Lotus Domino, accessing clients are faced with two potential login

points. Therefore, WebSEAL supports a single sign-on solution to one or more IBM WebSphere or Lotus Domino servers across WebSEAL junctions.

WebSphere provides the cookie-based lightweight third-party authentication mechanism (LTPA). You can configure WebSEAL junctions 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 WebSEAL. Upon successful authentication, WebSEAL 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 WebSEAL and the WebSphere or Lotus Domino server.

WebSEAL 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, WebSEAL 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 9-9 on page 261](#) shows the available usage scenarios with the LTPA authentication.

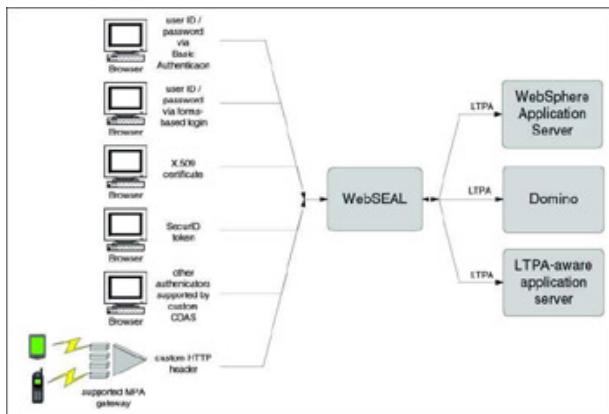


Figure 9-9: WebSEAL LTPA token delegation

- [1] From the security section of the Open Group Web site (<http://www.opengroup.org/security/topics.htm>) .
- [2] LDAP schema describes the way of storing the information in a LDAP directory in terms of object classes and attributes.
- [3] Note that a custom Web application should be developed in order to achieve the “on-the-fly” synchronization of main and GSO passwords. All password changes should be handled by users through that application that will subsequently carry out the changes for main and GSO passwords. This would enable main user passwords to be encrypted. However, the Access Manager mechanisms for setting up and maintaining password policies may not be in place any longer. Otherwise, the Access Manager native interface for changing passwords, and not the custom application, would be invoked (for example) in the case of password expiration.

Chapter 10: Access Manager Authorization

Overview

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, such as is shown in Figure 10-1 on [page 265](#):

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.

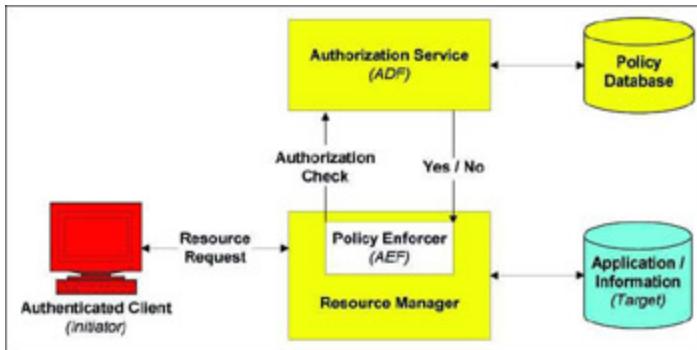


Figure 10-1: Open Group authorization

model 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.

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 databases, 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.

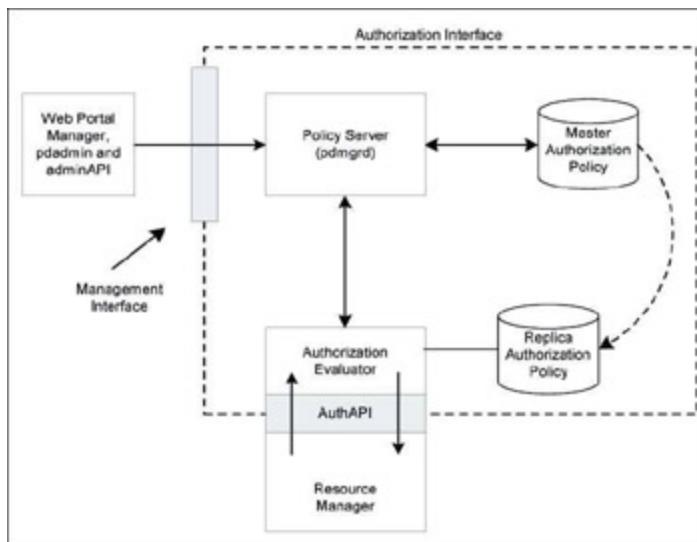


Figure 10-2: Access Manager authorization service

The diagrams in Figure 10-1 on [page 265](#) and Figure 10-2 clearly show how the Tivoli Access Manager authorization model maps to the ISO 10181-3 security architecture standard, that is:

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.

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, “Introduction to Access Manager components” on page 127](#).

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.

[[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>

10.2 Domains

Access Manager uses the concept of *domains* to manage a defined user base's access to protected resources. Access Manager supports the definition of multiple domains within one Access Manager installation.

A domain consists of all of the resources that require protection, along with the associated security policy used to protect those resources. These resources can be any physical or logical entity, including objects such as files, directories, Web resources, printer and network services, and message queues. Any security policy implemented in a domain affects only those objects in that domain. Users with authority to perform tasks in one domain do not necessarily have authority to perform those tasks in other domains.

In large enterprises you might want to define more than one domain. Each domain is given a name and is established with a unique set of physical and logical resources. The security administrator can define the resources in a domain based on geography, business unit, or major organizational division within the enterprise. The security policy defined in the domain affects only those objects within the domain, which enables data to be partitioned and managed completely independently.

Other benefits associated with multiple domains are as follows:

- Increased security

Security policy data for each domain is mutually exclusive. Users, groups, and resources that are defined within a domain cannot be associated with another domain.

- Simplified administration

You can assign independent administrators to handle policy management tasks for each domain.

An administrator assigned to a specific domain has authority only within that domain. However, by default, an administrator can view users and groups defined in the user registry that are not necessarily Tivoli Access Manager users or groups. This is beneficial if, for example, an administrator wants to import a user or group from a different domain.

10.2.1 Multiple domain model

Each Access Manager domain has its own master authorization database. A master authorization database is created when a new domain is defined. The database is then managed exclusively by the new domain. It contains the objects for the domain as well as ACLs, POPs, authorization rules, and Access Manager server definitions.

Each domain has its own Access Manager servers that use the ACL database defined for the domain. This includes WebSEALs, Authorization Servers, and any other applications using the C or JAVA authorization APIs.

Each domain has its own management users and groups. These are created when the domain is defined and are given the appropriate permissions to manage the domain ACL database, manage users and groups for the domain, and register Access Manager applications.

GSO information is also managed independently for each domain.

Each domain is responsible for managing its own users and groups. However, it is possible (but not required) for domains to share users because all domains share the same user registry.

All administration actions (using the pdadmin command line, WPM, and the administration APIs) require a domain to be specified. An administration tool can manage any domain (depending on the user who has logged in). This means that administration tools can be shared between domains.

All domains share a single set of Policy Server, Proxy Policy Servers, and user registry. Figure 10-3 depicts an overview of the components in a multi-domain implementation.

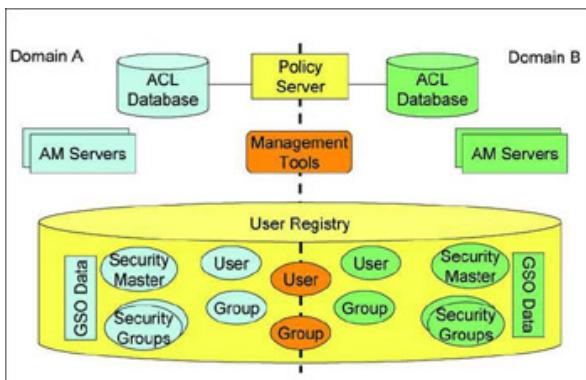


Figure 10-3: What is shared and separate per domain

Access Manager implements the concept of a *default* domain. In LDAP, for example, this was known by the name secAuthority=Default.

The multiple domain model still supports this concept. The default domain is the domain that is created when the Access Manager Policy Server is configured and is known as the Management Domain. The Policy Server is always registered in the Management Domain and other domains can only be configured by an administrator of the Management Domain. The Management Domain (through registry ACLs) has control of all Access Manager objects owned by all domains. This enables the Management Domain to delete domains and the objects that they own.

Access Manager servers can be registered into the Management Domain in the same way as they can be registered into other domains. In fact, using the Management Domain on its own creates an environment the same as in previous Access Manager configurations.

10.3 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 10-4 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.

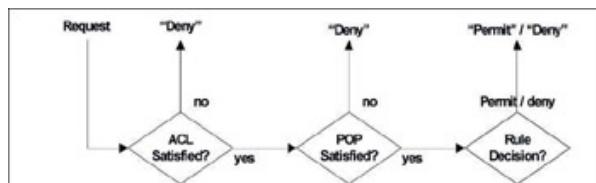


Figure 10-4: Authorization decision flow

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 275](#) 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.

10.3.1 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 10-5 on [page 274](#) shows a logical representation of a protected object space with multiple container and resource objects.

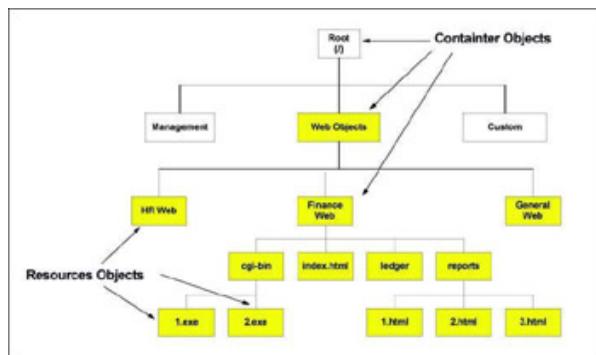


Figure 10-5: Access Manager protected object space

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.

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.

10.3.2 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.

10.3.3 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.

10.3.4 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.

10.3.5 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.

10.3.6 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 10-6 on [page 279](#) shows how the logical components of the rules engine are implemented using XML and XSL technology.

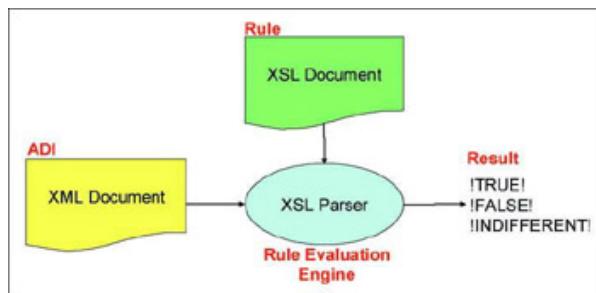


Figure 10-6: Authorization rules engine

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).

10.3.7 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 (ARS)

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 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 end 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.

10.4 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.

10.4.1 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.

Chapter 11: WebSphere 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 Java-based e-business Web applications, we can start to build a solution that enables us to distinguish and differentiate between security and application functions.

Looking at application development platforms within today's e-business environments, we have to take a closer look at Web application servers based on Java 2 Platform Enterprise Edition (J2EE). Two major products have already been integrated with Access Manager to some extent: the BEA WebLogic Application Server and the IBM WebSphere Application Server. This chapter examines the details of integration between Access Manager and WebSphere Application Server.

11.1 Business requirements

Security is such a fundamental enabler of e-business that in the emerging B2C and B2B markets, effective security can make the difference between owning the market and 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. That promise evaporates if security issues are not addressed properly.

In this world of e-business, WebSphere has become a Web application server market leader. A growing number of customers are deploying WebSphere and WebSphere-based solutions as their core e-business software platform. Few, if any, customers will put their WebSphere-based applications and solutions into production without the assurance that their business processes and data will be protected from malicious and inadvertent loss. More important, 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, including IBM WebSphere, 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 Web 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 Web 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](#).

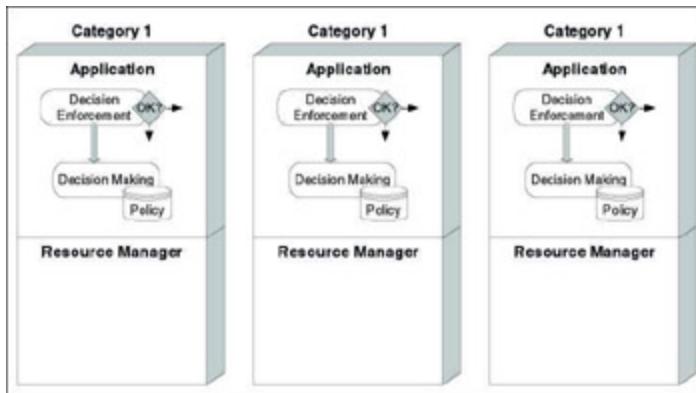


Figure 11-1: Category 1 systems

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.

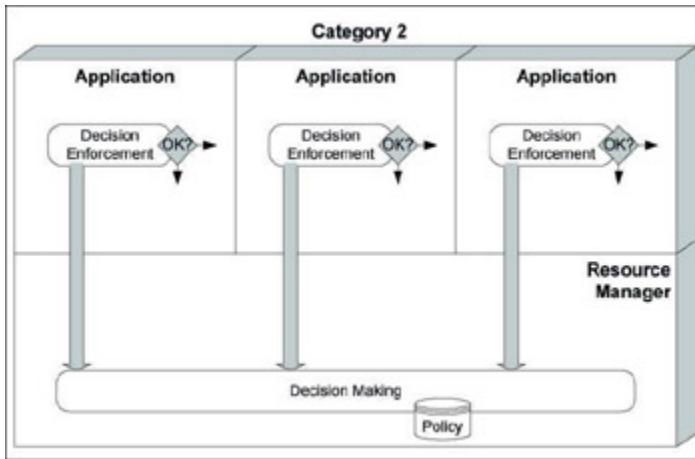


Figure 11-2: Category 2 systems

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](#).

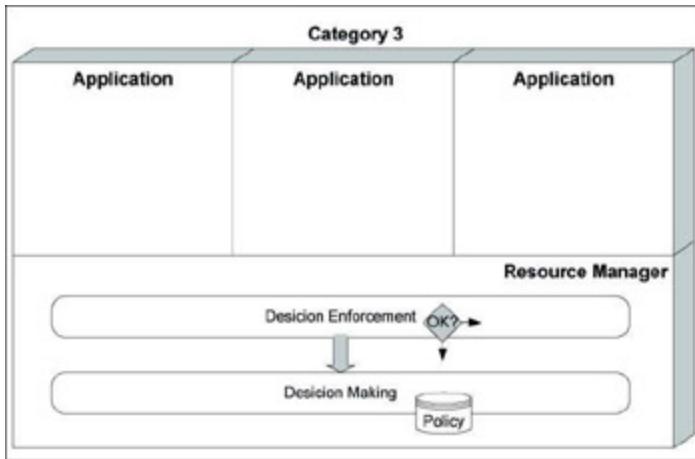


Figure 11-3: Category 3 systems

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 WebSphere Enterprise Java Bean (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.

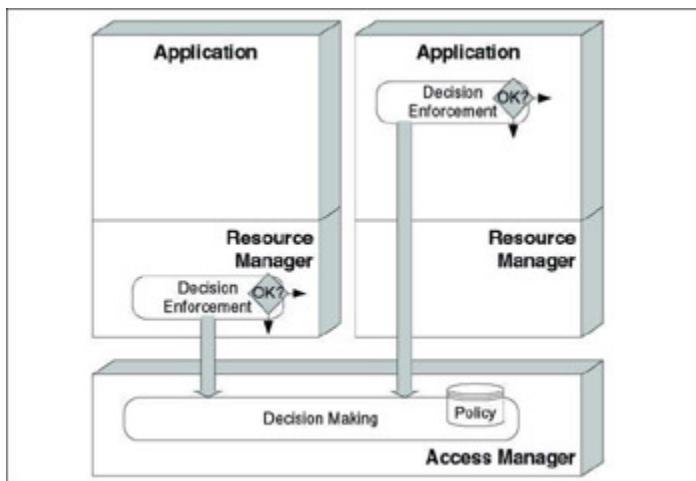


Figure 11-4: Policy enforcement based on consistent decision making

11.3 WebSphere Application Server security

WebSphere Application Server Version 5.1 is a Java 2 Enterprise Edition (J2EE 1.3) compliant Java application server that supports the Software Development Kit for Java Technology Edition 1.4. It implements the required services as they are specified. All components that are intended to run in WebSphere are packaged as Enterprise Archive files (EAR). It uses the Application Assembly Tool (AAT) to build an EAR from component modules and configure all deployment descriptors.

We concentrate on the J2EE security features implemented in WebSphere V5.1:

- Role-based security
- Declarative security
- Programmatic security

Role-based security

One of the goals of the EJB 1.1 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 287](#).

EJB 1.1 and J2EE 1.3 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 model has changed from permission-based to role-based.

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 Windows 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.

Listing 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](#) 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.

Listing 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 11-5](#) 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.

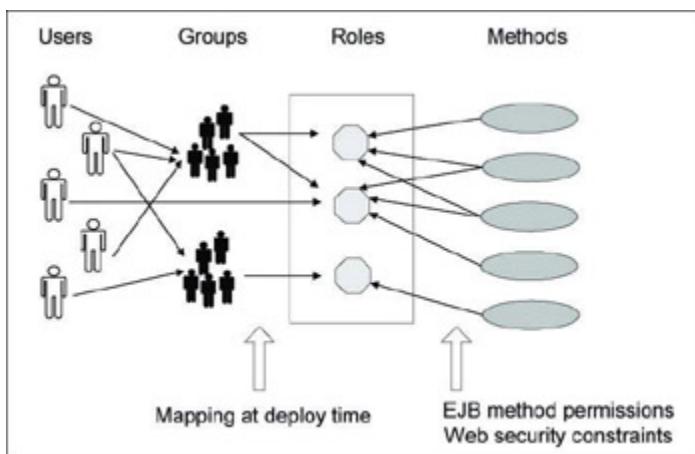


Figure 11-5: Role-based security

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 1.2 consists of two methods of the 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 rather than the permission-based model of previous versions. WebSphere Application Server treats a role as a set of permissions to access particular resources.

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
 - Web Trust Association Interceptor (TAI)
 - WebSEAL LTPA cookie support
- 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. [Table 11-1](#) provides an overview of supported user registries for WebSphere 5.1 and Access Manager 5.1.

Table 11-1: LDAP directories supported by WebSphere and Access Manager

LDAP Directory	WebSphere 5.1 support	Access Manager 5.1 support
Sun ONE Directory Server 5.2	X	X
Lotus Domino Enterprise Server 5 and 6	X	X

LDAP Directory	WebSphere 5.1 support	Access Manager 5.1 support
IBM Directory Server 4.1, 5.1, and 5.2	X	X
Windows Active Directory 2000 or 2003	X	X

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 **pdadmin** command and the WPM Administrator Console.

WebSphere never changes the default installation of the LDAP server, but Access Manager does. WebSphere and the LDAP server need additional configuration after Access Manager has been installed to enable all to work together. The changes to be aware of are:

- Anonymous access to LDAP is no longer permitted. WebSphere must be configured with a Bind Distinguished Name.
- Schema is modified. 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 must be able to perform

directory searches to retrieve users and groups and populate user and group-selection lists, so the WebSphere administration ID must be added to the LDAP Security group.

11.4.2 Single sign-on

Single sign-on between Access Manager and WebSphere can be achieved using three different mechanisms:

- GSO junctions
- Trust Association Interceptor
- LTPA cookies

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 end 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 Sign-On (GSO) lockbox” on [page 254](#).

Web Trust Association Interceptor (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 credentials of the authenticated user. WebSphere can then use these credentials to authorize the request.

Former versions of WebSphere did not understand the format of the credential information passed by WebSEAL. Since Version 3.5.3, WebSphere includes a new execution mode in its security framework, the Trust Association Interceptor Mode, in which it can interface with third-party objects that intercept 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. The method of validation is agreed on by WebSEAL and the interceptor. Setting values for parameters defined in the webseal.properties file that resides on the WebSphere

Application Server server determines the method of validation for the interceptors.

The TAI version that ships with Access Manager 3.9 can be configured in three different ways:

- Trust Association with -b supply option
- Trust Association without -b supply option (improved version)
- Trust Association using mutually authenticated SSL

In the following sections, we describe the different options and requirements.

Using TAI with a -b supply junction

In order to get TAI to work, there must be a WebSEAL junction to the WebSphere Application Server. In the earlier version of TAI, the only available authentication of WebSEAL was to secure the junction with the -b supply option, which has an associated risk for security attacks. The WebSEAL user ID and password used in TAI is not authenticated by WebSEAL, but is used by WebSphere to authenticate the traffic coming from WebSEAL. The following steps, as depicted in [Figure 11-6](#) on [page 298](#), explain this in more detail:

- When a user requests access to a WebSphere-protected resource through a browser, WebSEAL authenticates the user, generates the Extended Privilege Attribute Certificate (EPAC) (based on userid=John), and obtains his own WebSEAL user ID/password (ws_itso/chuy5) from the pd.conf file located on the WebSEAL server.

- WebSEAL sends this WebSEAL user ID and password in the BA header along with the authenticated user credentials in the iv-user, iv-groups, and iv-creds fields.
- WebSphere extracts the WebSEAL user ID/password in the BA header to bind to LDAP in order to authenticate the traffic from WebSEAL.
- The user ID in the iv-user header is extracted by WebSphere and used to authorize the request to access the protected resource.
- Because WebSphere trusts the TAI authentication over the junction (-b supply), the password for the user requesting access is not required or used by WebSphere.

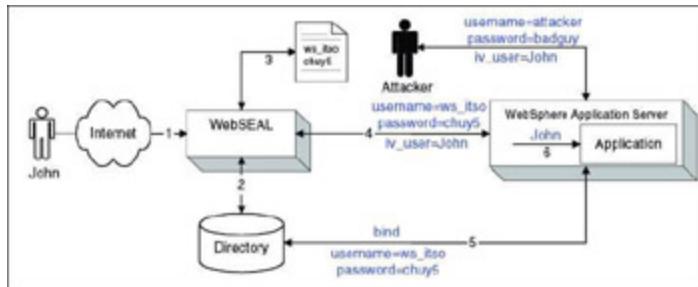


Figure 11-6: TAI using a junction with the -b supply option

Suppose now that we have an attacker who is a legitimate user of some other part of the system. This attacker can impersonate any other user by creating a packet with his own user ID and password in the BA header and any other user's ID in the iv-user header. WebSphere will bind to LDAP and use the attacker's ID to authenticate the traffic. The user ID in the iv-user header is used to authorize access to a protected WebSphere resource. This is not a good situation if the protected resource is, for example, a payroll system.

Using TAI without a -b supply junction

The new and current version of TAI provides better security. Only WebSEAL's password is sent in the BA header, which is being defined as the basic-dummy-passwd value in the pd.conf file located on the WebSEAL server. WebSphere uses the password sent in the BA header and then extracts WebSEAL's UserID value from the webseal.properties file located on the WebSphere Server to authenticate the traffic from WebSEAL.

Note The

com.ibm.websphere.security.WebSEAL.username value in the webseal.properties file contains the WebSEAL UserID.

Attackers who are legitimate users of some other part of the system would no longer be able to use their own user ID and password to impersonate a valid user. They would need access to the WebSEAL password. This version of TAI is shown in [Figure 11-7](#).

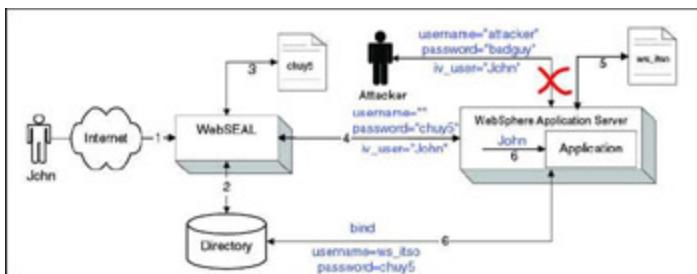


Figure 11-7: TAI using a junction without the -b supply option

Where an internal network is defined as secure, TAI without the -b supply option can be used without SSL.

Using TAI with mutually authenticated SSL junction

This approach improves the performance by eliminating the necessity of WebSphere calling LDAP for authentication of each WebSEAL connection. The `webseal.properties` file is located on the WebSphere Application Server and by setting the value of the `com.ibm.websphere.security.WebSEAL.mutualSSL` to yes, we can eliminate a BIND to LDAP to validate WebSEAL's identity. It can only be used when the junction between WebSEAL and WebSphere is a *mutually authenticated* SSL junction, as described in the section "Mutually Authenticated SSL Junctions" in the *Access Manager WebSEAL Administration Guide*. WebSphere will *trust* this junction and no additional mechanism is used to authenticate WebSEAL. This situation is depicted in [Figure 11-8 on page 300](#).

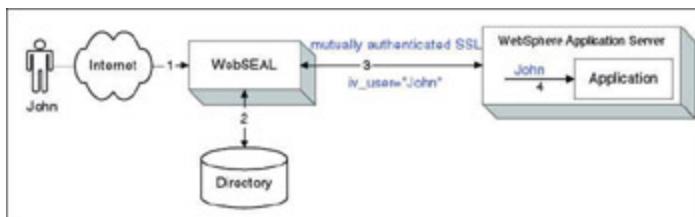


Figure 11-8: TAI using a mutually authenticated SSL junction

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 properly authentication, WebSEAL forwards a modified HTTP request to the back-end WebSphere server.

5. Depending on the configuration:

- If the junction has been defined without -b supply, the modified HTTP request contains the basic-dummy-passwd value in the BA header field that is only used between WebSEAL and WebSphere. This password and the value of the com.ibm.websphere.security.WebSEAL.username property is used to bind to LDAP. If this bind is successful, WebSphere will trust this session and the values sent with the http headers (this is done in method: validateEstablishedTrust()).

Note WebSphere uses the method validateEstablishedTrust to extract the WebSEAL user ID and password from the BA header and binds to LDAP to authenticate WebSEAL. If this fails, an exception is thrown, and WebSphere denies access.

- If the junction has been defined with -b supply, the modified HTTP request contains the user ID and password value of WebSEAL in the BA header field and the requestor's user ID in the iv-user field. WebSphere binds to LDAP using the user ID and password specified in the BA header.
- Alternatively, if the junction between WebSEAL and WebSphere is a mutually

authenticated SSL junction and the property value of com.ibm.websphere.security.WebSEAL.mutualSSL is yes, WebSphere trusts the session and does not need to bind to LDAP.

6. TAI then extracts the value of the iv-user http header and returns this as the authenticated user that should be used by WebSphere authorization. (This is done in method: getAuthenticatedUsername().)

WebSEAL LTPA cookie support

WebSphere Application Server uses an LTPA Token (a cookie by another name) to provide single sign-on across multiple WebSphere servers. After the user has been authenticated by WebSphere, an LTPA cookie is created and sent to the browser. The browser will return this cookie on subsequent requests, enabling the origin WebSphere Application Server (or other WebSphere Application Server within the same TCP domain) to recognize the user. The LTPA cookie is protected by a 3DES key, which also proves that it was created by a trusted source.

Access Manager WebSEAL can generate an LTPA cookie that will be accepted by the WebSphere Server. The WebSphere server trusts this LTPA cookie because it is encrypted with the correct shared key.

Figure 11-9 shows a WebSEAL junction that is configured for LTPA. An LTPA key is generated on the WebSphere machine and then exported to a keyfile (protected by a password). This keyfile must then be manually copied to the WebSEAL machine. When the junction to the WebSphere server is configured, it is specified as an LTPA junction, and the keyfile and password are given as parameters.

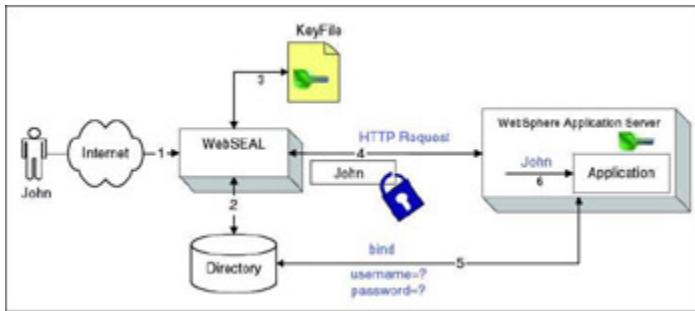


Figure 11-9: WebSEAL creates LTPA cookies for authenticated users

When a user authenticates to WebSEAL and requests a resource on this junction, WebSEAL creates an LTPA cookie using the key from the keyfile. This encrypted cookie contains the user's Access Manager UserID and is included in the HTTP request sent to the WebSphere server. When WebSphere receives the HTTP request, it extracts the UserID from the LTPA cookie and uses it to build a WebSphere credential for the user. It would be usual for WebSEAL and WebSphere to share a registry to avoid synchronization problems. WebSphere then applies its own authorization decision to the request. A general description of LTPA is provided in 9.5.6, "Using LTPA authentication with WebSEAL" on [page 259](#).

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 either the Trust Association Interceptor (TAI) or the LTPA support would be the preferred solution. GSO is only an option for the following scenarios:

- When WebSEAL and WebSphere rely on different user registries, 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.

- There might be situations, even in the case of a shared user registry, where -b gso might be useful. For example, if internal users should be able to connect to WebSphere directly using basic authentication and then have indirect access through WebSEAL with WebSEAL being configured to provide forms-based logon.

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 Application integration

If we want to integrate WebSphere applications with Tivoli Access Manager we have to distinguish between:

- Integration of new applications that are to be developed or existing applications that will be changed
- Integration of existing applications without any changes

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. These methods can be used for securing new applications or to adjust existing applications. We provide an overview of these methods in the next subsections, “aznAPI” and “*PDPermission* and JAAS.”

Often, there are already existing J2EE applications secured by WebSphere declarative security and/or using J2EE security methods. Tivoli Access Manager for WebSphere Application Server offers the possibility of importing WebSphere security definitions into Access Manager's object space. The function that determines whether a user is granted any permitted roles is now handled by Access Manager. We describe Tivoli Access Manager for WebSphere Application Server in "Access Manager for WebSphere Application Server" on [page 304](#).

aznAPI

aznAPI is an API specifically designed for Access Manager. It has been approved by the OpenGroup as the standard implementation of the Authorization Model. Access Manager provides a C version of the API, and Java wrappers are available as Open Source. WebSphere applications may use the aznAPI to retrieve fine-grained authorization information about a user.

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 287](#). 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.

Today, JSPs, servlets, and EJBs can take direct advantage of these services. When WebSphere containers support Java 2 security, EJB developers can avoid the need to make security calls by having the containers handle security while they focus on business logic.

There are two options for implementing fine-grained authorization (at the level of actions on objects) within servlets and EJBs today:

- Given the Access Manager credential information (EPAC) passed in the HTTP header, the servlet or the EJB would have to use the PDPermission class extensions directly to query Access Manager for access decisions. The access enforcement is still the responsibility of the application (servlet or EJB).
- Develop a proxy bean (a session bean) within an EJB. This proxy bean will intercept all method invocations and communicate with Access Manager (using the PDPermission class) to obtain the access decision and enforce it.

Access Manager for WebSphere Application Server

If the application is designed as a J2EE application, it would rely on the J2EE security methods to get a user ID and role. Tivoli Access Manager for WebSphere Application Server provides container-based authorization and centralized

policy management for WebSphere Application Server Version 5.1. Tivoli Access Manager for WebSphere Application Server is implemented as an Access Manager aznAPI application running on the WebSphere Application Server instance.

Note WebSphere Application Server Version 5.1 comes packaged with Tivoli Access Manager for WebSphere. No separate installation of Tivoli Access Manager for WebSphere is required for customers using WebSphere Application Server V5.1.

A graphical user interface utility, the Access Manager Web Portal Manager, and the **pdadmin** command provide a single point for security management of common identities, user profiles, and authorization mechanisms.

Access Manager for WebSphere Application Server supports applications that use the J2EE Security Classes without requiring any coding or deployment changes to the applications.

Tivoli Access Manager for WebSphere Application Server is used to evaluate access requests from a user to protected resources based on the following tasks:

- Authentication of the user
- Determination of whether the user has been granted the required role by examining the WebSphere deployment descriptor
- The WebSphere container using Access Manager to perform role membership checks for security code added directly into an application (programmatic security)

[Figure 11-10 on page 306](#) shows the integration of Access Manager with WebSphere Application Server Advanced Edition Version 5.1. When a user tries to access a protected resource by running an application with J2EE security, the following sequence of events occurs:

1. WebSphere Application Server authenticates the user. For WebSphere Advanced Edition, the authentication is against the IBM Tivoli Directory Server user registry. This LDAP user registry is shared with Access Manager.
2. WebSphere container collects information from the application deployment descriptor to determine the required role membership for the protected resource.
3. The WebSphere container uses the integrated Access Manager module to request an authorization decision from the Access Manager authorization server.
4. The Authorization Server obtains the user credentials from the Directory Server LDAP user registry and checks the user's permissions for a specified role. The Access Manager security model uses the definitions stored in the protected object namespace to build a hierarchy of resources to which ACLs can be attached. These ACLs define the mapping of roles in the J2EE application deployment descriptor to users and groups defined in the LDAP user registry.
5. The Authorization Server returns the access decision to the WebSphere container, which either grants or denies access to the protected resource.

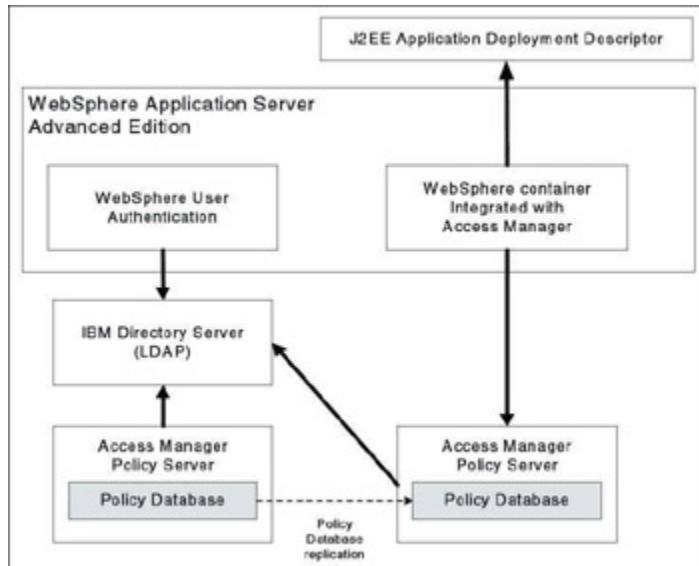


Figure 11-10: Access Manager integration with WebSphere Application Server

Migration of roles to principals and groups

Tivoli Access Manager for WebSphere Application Server has a migration utility that maps the roles in WebSphere Application Server to Access Manager principals and groups.

The *IBM Tivoli Access Manager for e-business IBM WebSphere Application Server Integration Guide Version 5.1*, SC32-1368, provides a fully detailed chapter about this migration utility and is recommended for further study. In this chapter, we give a brief description of the migration utility.

[Figure 11-11](#) on [page 307](#) depicts the single steps executed by the migration utility. This utility extracts information about roles from the J2EE deployment descriptor and maps them to group/principal information in Access Manager.

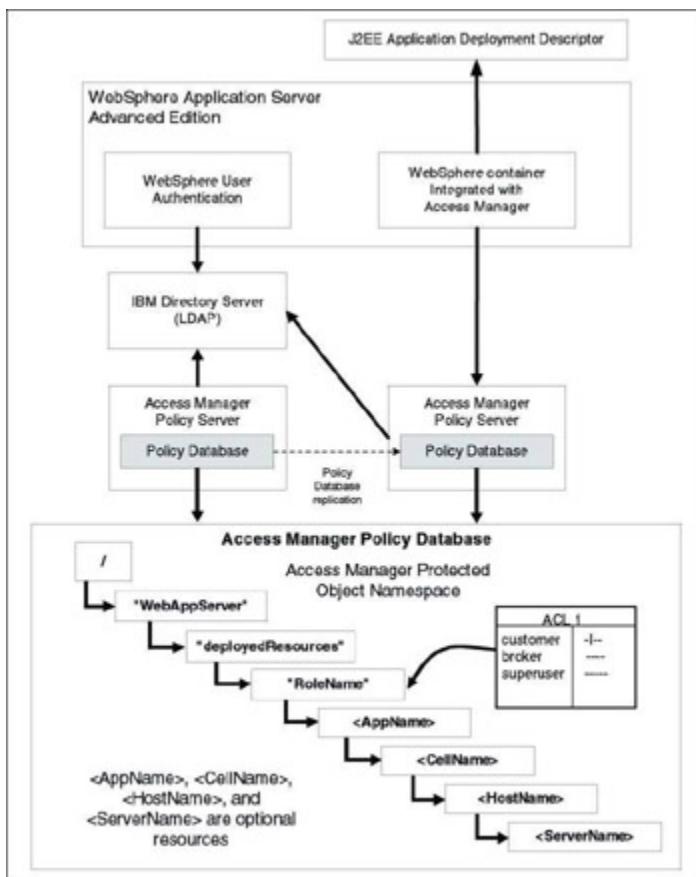


Figure 11-11: Access Manager utility to map roles to principals and groups

This information is converted into Access Manager format and is passed to the Access Manager Policy Server using the administration API, which is used to add entries to the protected object namespace to represent roles defined for the application. The appropriate principals and groups are added to the ACLs that are attached to the new objects. This migration utility is executed on the machine that has access to the J2EE application deployment descriptors (EAR files).

After migration, the system administrator can use the pdadmin commands or the Access Manager Web Portal Manager to modify the ACLs and add new roles and users.

Attention The migration utility has the following limitations:

- The migration utility is designed only to migrate the roles in EAR files to the Tivoli Access Manager protected object space. Do not use the migration utility as a maintenance utility for roles. After migrating an EAR file, use either the Web Portal Manager or the pdadmin utility to manage roles.
- The migration utility migrates only the user and roles specified in the EAR file. Ensure that you are using the latest EAR file for your application.
- After the migration utility has been run once against an EAR file, it is recommended that you do not run it again when changes are made to an EAR file. The following problems can occur when an EAR is created and migrated to the protected object space, and then is migrated again.
 - On the second or subsequent migrations, if an existing role has been removed from the EAR, it will not be removed from the protected object space.
 - On the second or subsequent migrations, changes to the EAR file might require the migration utility to instruct Tivoli Access Manager to delete an ACL definition. In some cases, Tivoli Access Manager may prevent

this deletion. Note that the migration of an EAR file to the Tivoli Access Manager-protected object space results in the creation of ACLs that are attached to objects. If the administrator has manually attached the ACL definition to other protected objects, Tivoli Access Manager prevents removal of the ACL. Thus, even if the original object that was created by the first run of the migration utility no longer exists, the ACL cannot be deleted.

- Use pdadmin to modify roles. You can use pdadmin to add additional roles.
- When using the migration utility with WebSphere Application Server Advanced Edition Single System Edition, you must manually add the users to the ACLs that were created by the migration utility. This limitation does not affect WebSphere Application Server Advanced Edition.

Central policy management for multiple WebSphere servers

Tivoli Access Manager can manage security policy across multiple WebSphere servers.

After using the migration utility as described earlier, the Access Manager Web Portal Manager can be used to

manage changes in security definitions related to the mapping of roles to principals and groups.

Important Changes to role mapping made through the WebSphere console will not be visible in the Access Manager security model.

The Access Manager security model may also be managed using the **pdadmin** commands. Access Manager also provides a programmatic interface to administration tasks accomplished by **pdadmin** and the Web Portal Manager. The APIs can be used by programmers to develop tasks specific to an application.

In our security architecture, we have chosen to use multiple WebSphere servers to host a new Internet application. This is a high availability consideration; Server 2 is a mirror copy of Server 1. Each server accepts Web traffic simultaneously routed to it through the load balancing capabilities of WebSeal. Central administration of the security policy is required to achieve this.

[Figure 11-12](#) illustrates central administration of two WebSphere servers. The Access Manager Web Portal Manager has been installed on *machine A*. Access Manager Policy Server with the master policy database is installed on *machine B*. The Web Portal Manager uses the Access Manager Policy Server to administer the security policy.

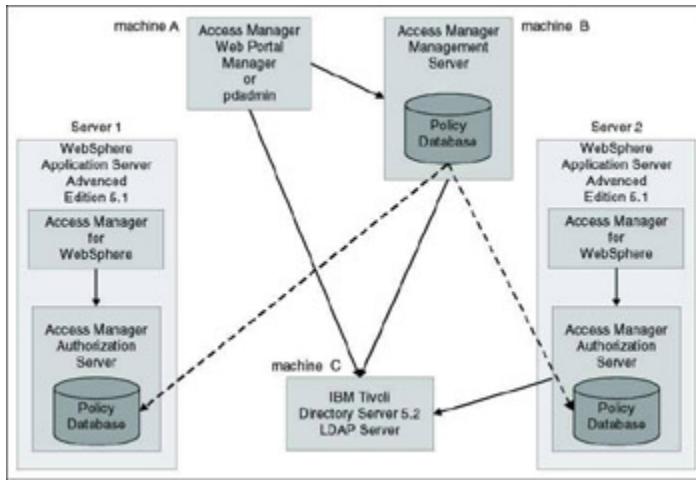


Figure 11-12: Central administration for multiple WebSphere servers

Two WebSphere Advanced Edition machines are shown: *server 1* and *server 2*. Access Manager Authorization Server is also installed on these machines. Co-location of Access Manager Authorization Server with WebSphere Application Server optimizes performance when making authorization decisions. This is because Tivoli Access Manager for WebSphere Application Server is an aznAPI application, and it communicates with the Authorization Server. Therefore, placing the Authorization Server on the same machine as the WebSphere Application Server application eliminates network connection overhead. This configuration is recommended.

The Access Manager policy databases on *server 1* and *server 2* are replicated from machine B, which improves performance and fail-over capability.

Access Manager servers and the WebSphere servers share the same LDAP user registry (in this case, IBM Tivoli Directory Server Version 5.2).

Access Manager provides a powerful command-line based utility (**pdadmin**) for the management of the Access Manager

security domain. Administrators can also use this utility within scripts or programs to automate administration tasks.

Access Manager also provides a programmatic interface to the administration tasks accomplished by **pdadmin** and the Web Portal Manager. Application developers can use this API to perform administration tasks that are specific to the application.

For more information, see the *IBM Tivoli Access Manager for e-business Administration C API Developer Reference Version 5.1*, SC32-1357, or the *IBM Tivoli Access Manager Administration Java Classes Developer Reference Version 5.1*, SC32-1356.

Chapter 12: Access Control in a Distributed Environment

To this point our discussions of Access Manager have focused primarily on WebSEAL scenarios in which all components are typically deployed in a relatively contained single security domain deployment environment at a single site. This chapter addresses key issues relating to situations where components of both single and multiple Access Manager secure domains may be distributed among multiple sites. We also touch on the use of Access Manager for Business Integration as a component of a distributed application.

12.1 Cross-site distribution of a single security domain

In order to introduce the aspects of multiple security domains, we take a closer look at the distributed Web security requirements at Stocks-4u.com.

Earlier, in [Chapter 7, “A basic WebSEAL scenario” on page 201](#), we introduced Stocks-4u.com’s key Web security requirements. Our scenario discussion focused on supporting a Web security infrastructure at a single site: the San Diego IT Center. We mentioned the need for future expansion of Internet operations into the Savannah center to provide for a redundant access capability with load balancing for customers on the US East and West coasts. After the initial Access Manager deployment in San Diego, Stocks-4u.com management has initiated planning for an extension of the security infrastructure to the Savannah IT Center, starting with a review of key business and technical requirements.

12.1.1 Business requirements

The need to distribute Web security is related to some key business issues:

- The growth rate of the customer base will require Internet application capacity to double over the coming year and double again the following year. The San Diego site will require expansion to support this growth. However, Savannah has substantial additional space that, if utilized, could reduce the San Diego expansion need by 70%.
- Stocks-4u.com competitors have been promoting higher service availability levels. Competitive pressures require the company support equivalent capability. A recent partial site power failure at the San Diego IT center demonstrated that expected availability levels cannot be achieved by simply replicating Access Manager servers at a single site.
- Related to the above, as part of its premium service offering, Stocks-4u.com has recently established service level

agreements in place with a number of its key customers. Should it fail to achieve service level targets, these customers are entitled to substantial levels of compensation.

- The majority of internal applications are deployed in Savannah. To utilize Access Manager security capabilities, internal user browser access must be routed through a WebSEAL server in San Diego. However, 68% of company employees are connected to the company intranet through the Savannah IT Center. A consultant has found that 15% of the company's total long-haul telecom cost is associated with routing internal HTTP/HTTPS traffic through San Diego.

Based on these issues, the following business requirements have been proposed by management:

- Savannah facilities be fully utilized to minimize the San Diego expansion requirement.
- Internet customer application access must be immune to a San Diego site failure.
- Management has directed that cross-site telecom costs for internal applications be reduced by 25% and the savings be used to expand network support for customer applications.

12.1.2 Technical requirements

An analysis of Stocks-4u.com business requirements has established the following key technical requirements:

- Internet users must be able to securely access applications via a Web browser through either San Diego or Savannah.
- In the event of a site failure or shutdown, all security functions must be provided through the surviving site.
- Internal HTTP/HTTPS application traffic must be routed through the closest IT center (San Diego or Savannah).
- The Stocks-4u.com network must provide direct network linkages from San Diego to New York trading systems.

If these technical requirements are met, the business requirements will be easily satisfied, and Stocks-4u.com will also be well-positioned for future expansion.

Security architecture changes

Currently, Access Manager components are deployed only in San Diego. The updated architecture will add new Access Manager components in the Savannah IT Center. There will be a single security domain sharing a common user registry, as shown in [Figure 12-1 on page 314](#).

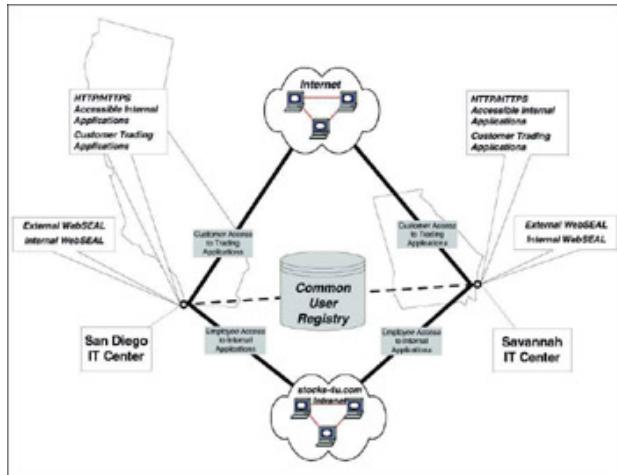


Figure 12-1: Stocks-4u.com security domain

Network changes

To support a distributed environment, the company has planned some changes to their network. Previously, customer trade transactions required routing through Savannah and ML&J in New York. A new T3 link is planned, which will directly connect San Diego to New York trading systems. Another link from the Internet to Savannah is planned to support direct access as customer application Web servers are deployed there. [Figure 12-2 on page 315](#) shows the new Stocks-4u.com corporate network connectivity.

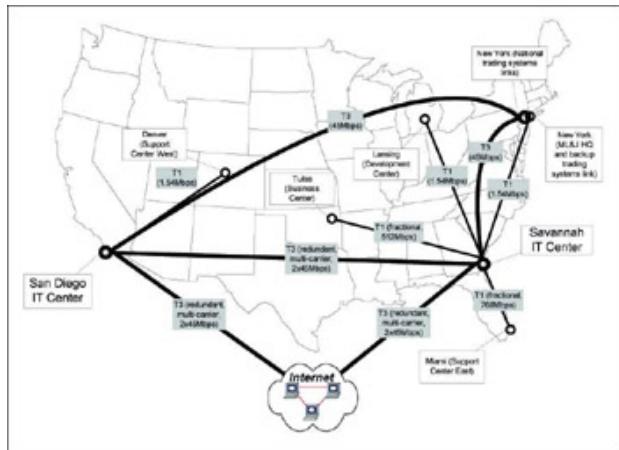


Figure 12-2: Stocks-4u.com updated data network

Within the Savannah IT Center, the internal network configuration will also change to support the new requirements. The new configuration will add an Internet demilitarized zone (DMZ), providing both IT centers with similar zone structure, as shown in [Figure 12-3 on page 316](#).

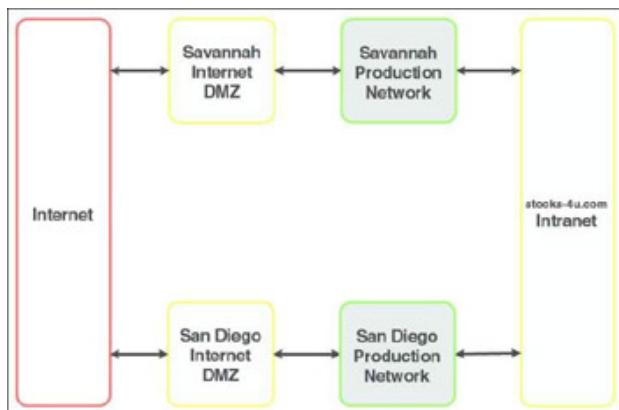


Figure 12-3: IT center network zones

It is important to note that communication between the IT center production zones must go through the corporate intranet.

12.1.3 Access Manager architecture discussion

In the initial WebSEAL deployment, all Access Manager components were installed in San Diego. In the distributed architecture, new Access Manager components must be deployed in Savannah. The

question is, which components? In this scenario, we must be concerned about:

- WebSEAL servers
- The Access Manager Policy Server
- The user registry
- The Web Portal Manager

WebSEAL servers

Certainly, we know that we will need to put WebSEAL servers in Savannah to avoid rerouting requests unnecessarily through San Diego. There will be both internal and external WebSEAL servers at each IT center. As in the existing Access Manager deployment, shown in Figure 7-4 on [page 210](#), the external WebSEAL servers will go into the Internet DMZs, and the internal WebSEAL servers will go in the production networks.

WebSEAL junctions

WebSEAL junctions may be cross-site (that is, WebSEAL servers in San Diego may be junctioned to back-end Web servers in Savannah and vice versa). This is not a problem as long as the cross-site communication between WebSEAL and the junctions is appropriately secured.

For external (Internet-facing) WebSEAL servers, another issue must be addressed when junctioning cross-site: An appropriate network configuration must be created to permit them to pass traffic from their respective DMZs into the production network at the remote site. This obviously is a more complex network scenario than the local site case, and we will discuss some of these networking issues.

As an alternative, Stocks-4u.com may choose to replicate all Web servers between sites so that all WebSEAL junctions are local. [Figure 12-4 on page 318](#) illustrates both approaches.

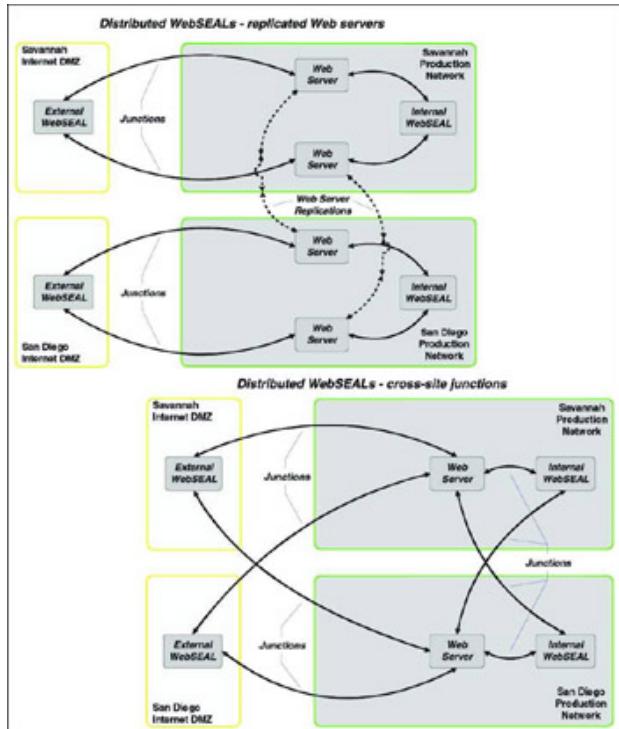


Figure 12-4: Distributed WebSEALs: replicated Web servers

Access Manager Policy Server

Because there can be only one Access Manager Policy Server for a security domain, the server in San Diego will remain in place and will support both sites.

In the event of a San Diego site outage, Savannah WebSEAL servers will continue to operate. However, configuration changes cannot be made. To address an extended site outage situation, a backup system will be available to take over the Policy Server function. Configuration of a replacement Policy Server involves special considerations.

Cross-site backup takeover considerations

Availability issues are often more involved when dealing with migration of functions across site locations. In a local fail-over situation it is much easier to utilize specialized high-availability software (such as HACMP or Microsoft Cluster Server) to provide an automated recovery capability. Such products generally utilize special network/device connectivity and *heartbeating* to determine

failure conditions and initiate a service takeover. Generally, a shared disk is used to assure data consistency

Consider that in a cross-site failover scenario, the network subnet addressing may be different between the sites, depending on the particular network architecture. This means that a simple IP address takeover may not be sufficient to bring a backup system online.

In the case where the backup has a different IP address, services that access it must be told of the new address, either by reconfiguration or via DNS changes. (To successfully accomplish an address change via DNS, services must not be caching the old address.)

In the alternative, a *shadow* standby network may be created at each site. These shadow networks are configured to the same subnet as the peer site and are kept offline unless a takeover is required. The backup systems reside in this network. Another twist to this, which would permit the shadow network to be accessible at all times, would be to use Network Address Translation (NAT) in front of the shadow network during normal operation. This would permit network traffic to flow to the backup systems (for example, capturing of configuration updates, and so on). Then, should a takeover be required, the router would have to be removed in order for the backup systems to appear on the local network under the peer site subnet. This approach, of course, assumes a total peer site network failure or loss of communication, because the peer network cannot be accessible simultaneously with the shadow using the same subnet address.

Another problem is that of replicating the existing configuration. Generally for Access Manager, this can be handled sufficiently by doing a periodic backup of critical files and replicating them to the backup site. (There are also more sophisticated mechanisms that can capture updates at the device level and replicate changes across sites.)

Perhaps the most difficult issue of all, however, is that of how to deal with the following:

- How can a failure of the primary server be reliably determined across site locations?

- If a backup server takes over, what happens when the primary comes back up?

These issues are very important because unpleasant scenarios can occur if a backup takeover should occur by mistake or without appropriate advance process planning—especially if it occurs when the primary is still actually active (which creates what is known as a *split-brain* scenario).

A communication failure across sites is not sufficient to determine with certainty that the primary has failed. After a backup takes over Policy Server functionality, the primary cannot be permitted to return to full operation without assurances that updates that may have occurred in the interim have been reconciled.

In addition to a manual process that may be executed to bring a backup server online in a cross-site situation, there are products that provide sophisticated cross-site fail-over capabilities. They are beyond the scope of this book.

The user registry

In the existing San Diego configuration, there is a master LDAP server for the user registry and an LDAP replica of the registry that is used by WebSEAL for authentication. In our distributed configuration, we place an additional replica in the Savannah production network. This replica will be used by the Savannah WebSEAL servers for user authentication, with the San Diego replica as a backup. The master remains in San Diego.

In this configuration, if the San Diego site goes down, the Savannah WebSEAL servers will still have registry services available for authentication. However, updates go through the San Diego master. In the event of an extended San Diego outage, a backup replica server will be *promoted* to a new master.

LDAP master promotion considerations: As with the Access Manager Policy Server, migrating a key function across site locations has special considerations. See “[Cross-site backup takeover considerations](#)” on [page 319](#) for further details.

Web Portal Manager

An additional instance of the Web Portal Manager will be installed in Savannah.

Distributed server configuration

[Figure 12-5](#) shows the distributed Access Manager server configuration.

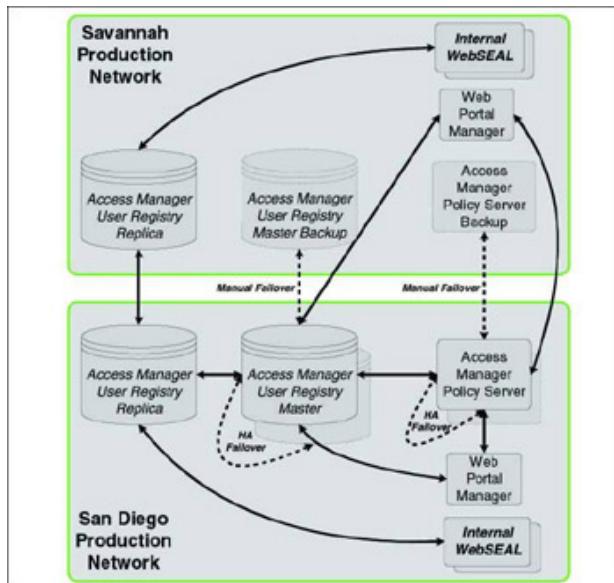


Figure 12-5: Stocks-4u.com production network distributed server configuration

12.1.4 Communication between distributed components

Although we have established where Access Manager components will be placed, we have not yet addressed cross-site network communication issues.

Cross-site component interactions

WebSEAL servers generally communicate with a local user registry instance for authentication. Cross-site communication issues occur with managing and updating configuration data. The main communication types required are as follows:

- The LDAP master in San Diego must communicate with its Savannah replica or replicas.

- The Policy Server in San Diego must also be accessible from the Savannah Web Portal Manager.
- The Policy Server in San Diego and the WebSEAL servers in Savannah must communicate with each other.

LDAP and Web Portal Manager traffic has to traverse whatever network infrastructure exists between the two sites. (The following discussion presumes three alternatives for this communication.) However, to support the Policy Server to WebSEAL communication, a Proxy Policy Server is introduced at the Savannah site. This increases security around the Policy Server because there are no direct connections from the remote site, and it increases ACL database replication time due to the caching capabilities within the Proxy Policy Server. [Figure 12-6](#) on [page 323](#) shows the communication flows.

Cross-site communication alternatives

To assure the integrity and privacy of cross-site traffic between security components, we consider three approaches:

- Using SSL across the open Stocks-4u.com intranet
- Using VPN capabilities to tunnel between network zones
- Bridging of network zones

SSL

All Access Manager components support the ability to interact via SSL:

- The Access Manager Policy Server communicates with WebSEAL and other blades via SSL.
- The Web Portal Manager communicates with the Access Manager Policy Server via SSL.
- The LDAP user registry can be configured to communicate via SSL. (Other supported LDAP servers may support SSL as well.)
- WebSEAL may be configured to communicate with junctioned servers via SSL, provided the servers support it.

If all components can be configured to use SSL, then Stocks-4u.com cross-site communication may be implemented securely using the company intranet. This may be a viable approach, especially in situations where the amount of cross-site communication is relatively small and the SSL overhead remains low.

Figure 12-6 shows the use of SSL between Stocks-4u.com IT centers.

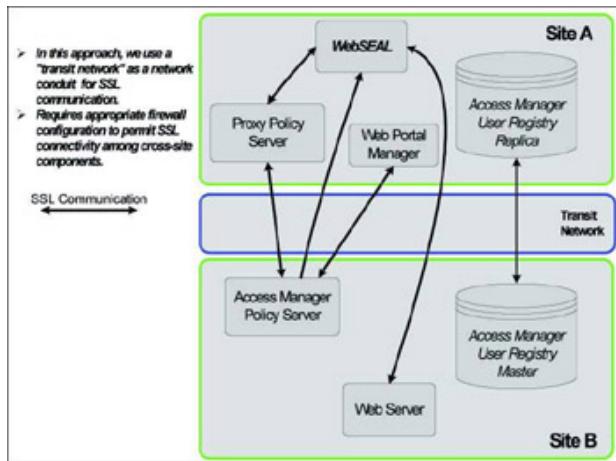


Figure 12-6: Cross-site SSL communication

A VPN approach

An alternative approach to cross-site communication involves the use of VPN technology to *tunnel* between the production networks. For example, a VPN tunnel can be created through the company intranet, which effectively permits unrestricted communication among the cross-site components without additional measures. The VPN traffic may be encrypted, providing for the integrity and privacy of communication between San Diego and Savannah without using SSL.

This approach has the advantage that local communication among Access Manager components does not have to incur encryption overhead as it may if SSL were used. Also, because the hardware/software cost of VPN technology has lowered, it is no longer cost-prohibitive in many cases. Figure 12-7 illustrates the use of a VPN for Stocks-4u.com cross-site Access Manager traffic.

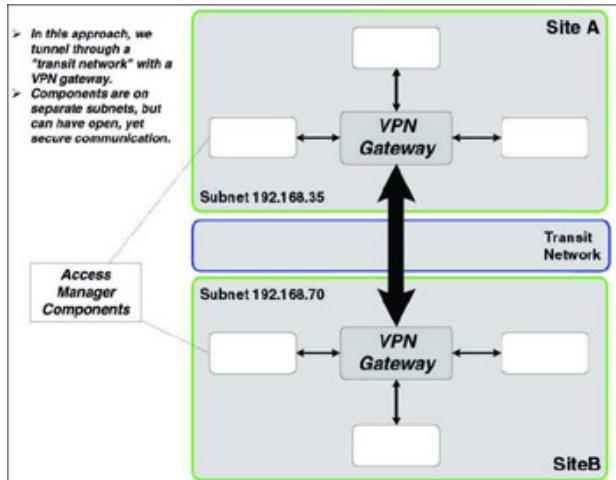


Figure 12-7: Cross-site VPN communication

Bridging production networks

A third approach may be to effectively *merge* the San Diego and Savannah production networks by *bridging* them. The two production networks may then co-exist within a single IP subnet. Current network hardware capabilities and costs may make this an attractive approach, especially because it may provide for cross-site IP address takeover when bringing up a backup Policy Server or LDAP master.

Note The ability to migrate IP addresses across sites does not eliminate the *split-brain* issue mentioned earlier. What it does do, however, is eliminate the need to reconfigure other systems to use alternate IP addresses for backup servers, or to create a *shadow* backup network at the site. In any case, caution and proper planning is still necessary in implementing any availability solution.

[Figure 12-8](#) illustrates the use of a bridged approach to distributing Access Manager components between San Diego and Savannah.

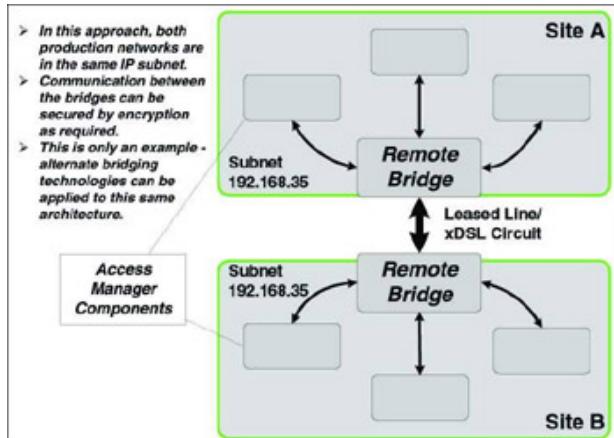


Figure 12-8: Bridged cross-site communication

12.1.5 Stocks-4u.com distributed architecture

Summarizing the architecture discussion, the Stocks-4u.com distributed Access Manager architecture will include the following:

- An architectural choice has been made to utilize a VPN tunnel to connect the Stocks-4u.com production networks together.
- WebSEAL hosts will junction across sites for internal applications. For external (customer) applications, Web servers will be replicated across sites and WebSEAL junctions will be local.
- Within the Savannah site a Proxy Policy Server will be used to support the local WebSEAL servers to avoid any direct connection from the WebSEAL servers to the Policy Server.
- Within the San Diego site, a standard high-availability configuration using HACMP will be used with the Access Manager Policy Server and the LDAP master. This will assure availability of services as long as the San Diego site remains operational.
- In the event of a communication disruption between San Diego and Savannah, operation will continue without configuration update capability at the Savannah IT Center.
- In the event of a confirmed San Diego site outage lasting more than eight hours, a backup Access Manager Policy

Server and a new LDAP master will be brought online in Savannah. (Strict procedures must be in place for this and for bringing the old servers back online upon site recovery.)

[Figure 12-9](#) on [page 327](#) summarizes the Stocks-4u.com distributed Access Manager architecture.

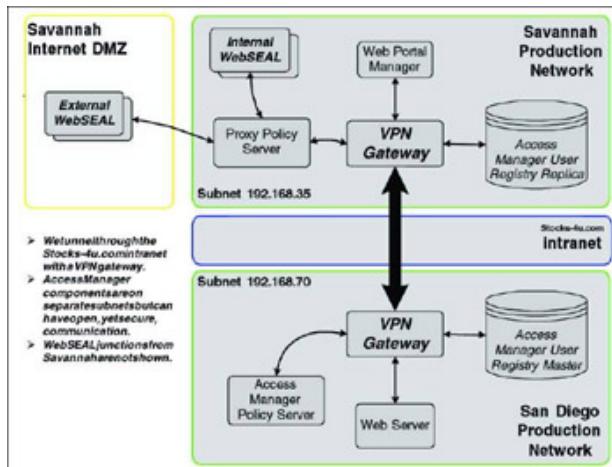


Figure 12-9: Stocks-4u.com distributed WebSEAL architecture summary

12.2 Distributed security domains

Another type of distributed scenario involves single sign-on for access to resources in multiple security domains (that is, each domain has its own user registry and security policy definitions). Consider two divisions of a company, each offering Web-based services to Internet customers. Each division has deployed WebSEAL in separate security domains. However, there is a requirement for certain users in one domain to access resources in the other domain without needing to authenticate twice. WebSEAL supports two different types of cross-domain authentication to address such scenarios: Cross Domain Single Sign-On (CDSSO) and e-community single sign-on.

12.2.1 CDSSO

WebSEAL supports the ability to forward an authenticated identity from a user in one security domain to a WebSEAL server in another security domain. The *receiving* WebSEAL then maps the identity provided by the *sending* WebSEAL to an identity that is valid in its security domain. CDSSO can also be viewed as a *push* model with respect to authentication.

This functionality is known as Cross Domain Single Sign-On. In CDSSO, the user makes a request to a special link on a WebSEAL server, which then initiates the process to forward the request, along with credential information to a WebSEAL server in a different Access Manager domain. If the user were to instead directly access the link in the target domain, he would have to authenticate to that domain.

The CDSSO process contains the following steps:

1. A user initially logs on to a WebSEAL server in one security domain.
2. At some point the user accesses a link controlled by the user's WebSEAL, which contains a special directive (pkmscdsso). This directive results in redirecting the user to a URL controlled by a WebSEAL server in another security domain and passing encrypted credential information to the new WebSEAL.
3. The user is redirected to the other WebSEAL and this server decrypts the credential information passed to it, maps the

identity to one defined in its own user registry, and then creates a secure session with the browser.

4. At this point the user has established secure sessions with two WebSEAL servers in different domains, but has only had to log in once.

Another way of looking at CDSSO is that it provides a mechanism by which a WebSEAL server in one security domain can send something analogous to a *letter of introduction* to a WebSEAL server in another security domain.

[Figure 12-10 on page 329](#) summarizes a typical CDSSO flow.

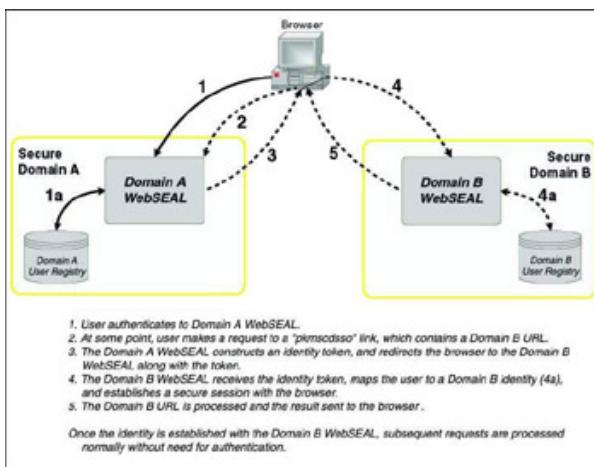


Figure 12-10: CDSSO identity determination process

The only significant CDSSO implication for a given security domain 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. The important thing to remember is that the mapping must make sense for the specific situation.

It is possible (using the CDMF interfaces discussed in 5.5.6, “Cross Domain Mapping Framework” on [page 163](#)) 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.

Using CDSSO at Stocks-4u.com

Let us briefly examine a possible CDSSO scenario at Stocks-4u.com.

Assume that ML&J has installed Access Manager in its New York offices to manage security for a new set of Web-based tools. This Access Manager domain is separate from the Stocks-4u.com domain and has its own user registry. Stocks-4u.com wants to make these tools available to certain ML&J customers without requiring them to log in twice (once to a Stocks-4u.com WebSEAL and then to an ML&J WebSEAL).

By setting up CDSSO between the Stocks-4u.com and ML&J Access Manager domains, users are required to authenticate only once. Also with CDSSO, initial authentication can take place in either domain, so that, for example, users initially authenticating to an ML&J WebSEAL server can access their Stocks-4u.com account.

Also, note that with CDSSO, an authorization decision can be made at the Stocks-4u.com WebSEAL to control participation in cross-domain single sign-on. The ability to do this may be important for business reasons.

[Figure 12-11](#) illustrates the described CDSSO scenario.

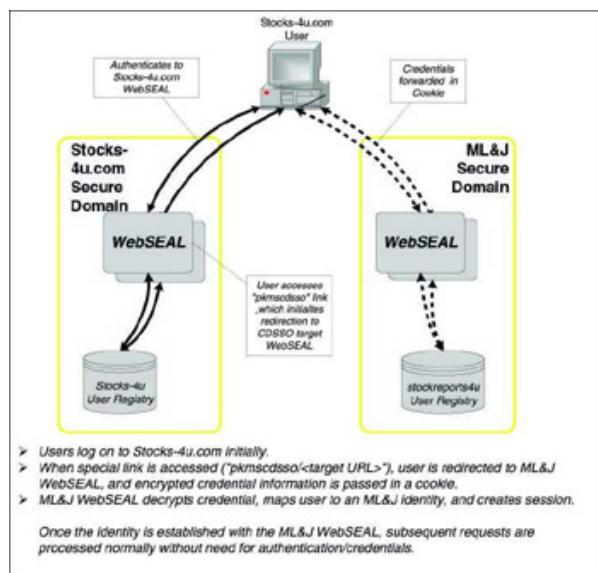


Figure 12-11: Stocks-4u.com CDSSO scenario

12.2.2 e-Community single sign-on

e-Community single sign-on supports a cross-domain authentication capability. However, it differs from CDSSO in a few key respects. 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.

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 has taken place, 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 WebSEAL servers in other domains within the e-community.

The e-community model is shown in [Figure 12-12](#).

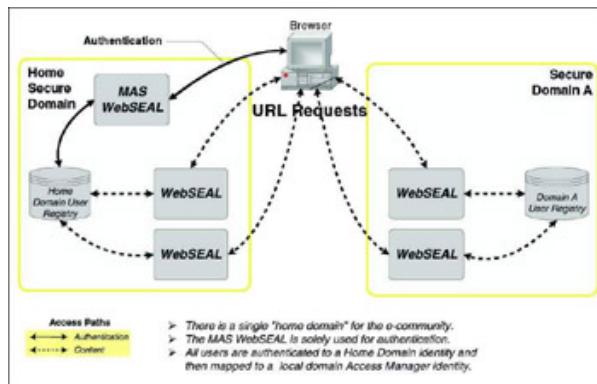


Figure 12-12: The e-community model

The e-community mechanism involves the following steps:

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 12-13](#) on [page 333](#) summarizes the flow of an initial e-community user authentication.

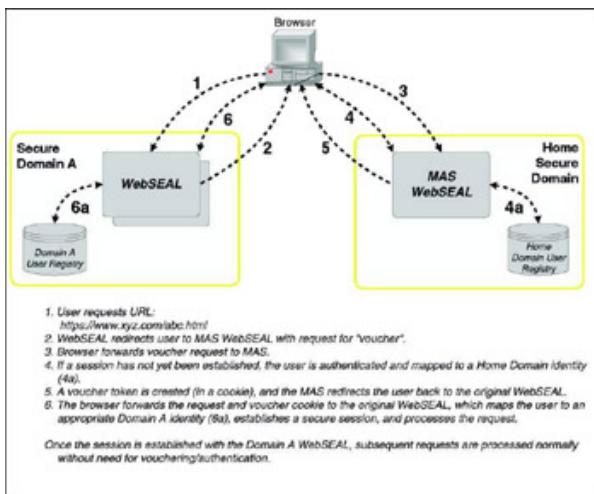


Figure 12-13: e-Community initial identity determination process

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 domain that validates the user's identity against the MAS then vouches for that user's identity within the local domain. This is depicted in [Figure 12-14](#) on [page 334](#).

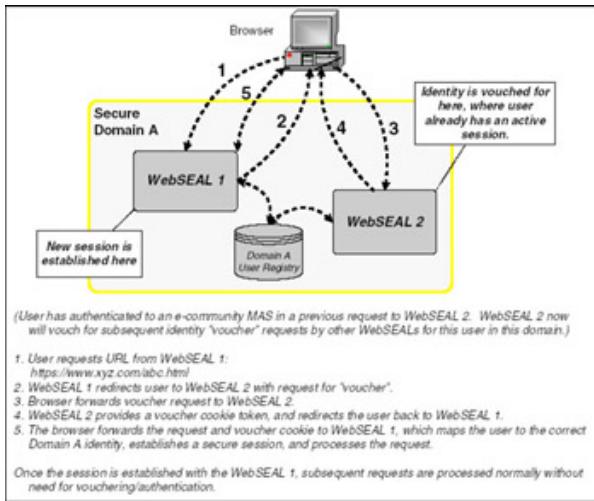


Figure 12-14: e-Community subsequent identity determination process

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.

There are many detailed issues regarding the operation of e-community single sign-on, but they are not architecturally important. The main architectural impact of e-community single sign-on involves the role of the MAS. The key issue is, with all user authentication for the e-community going through a single domain, where should the MAS server (or servers) be located?

This question is especially important in a geographically distributed situation. For example, consider the Stocks-4u.com distributed scenario.

Applying e-community single sign-on at Stocks-4u.com

Assume that Stocks-4u.com has just signed an agreement with a company that provides stock analysis reports via the Web. This company, stockreports4u.com, has an existing client base to which they charge individual usage fees. The Stocks-4u.com agreement

allows all Stocks-4u.com users to have access to stockreports4u.com through a special Web site for a flat fee (no individual usage charges).

Stockreports4u.com does not want to manage a user registry of valid Stocks-4u.com users. Instead, they wish to allow users to log in to their site against the Stocks-4u.com user registry, and then map those IDs to a small set of special IDs at their Web site.

Further, they wish to permit direct access to this site without a requirement to link to it through Stocks-4u.com first (that is, it may be directly *bookmarked*). In this scenario, depicted in [Figure 12-15](#) on [page 336](#), an e-community approach may be useful.

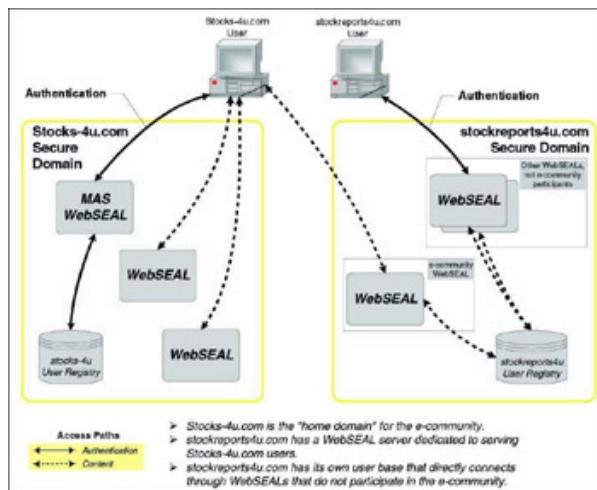


Figure 12-15: Stocks-4u.com e-community scenario

12.2.3 Comparing CDSSO and e-community single sign-on

It is difficult to state hard-and-fast rules regarding when it is best to use an e-community versus a CDSSO approach. Certainly one factor is the desired degree of control over the users.

In an e-community scenario, one site gets to control the users. Business models that are focused on profiling, branding, or other factors as mechanisms for revenue generation may be well-served by this approach.

Also, with e-Communities, if authentication at the MAS fails, users may have the option of authenticating directly to the local domain.

12.3 Conclusion

In this chapter, we have focused on more advanced Access Manager architectural issues that relate to its use in distributed environments. As you can see, there are a number of choices that may be required to complete an operational architecture in a distributed scenario.

This said, however, we hope that this chapter has shown that distributed architectures with Access Manager are not terribly complex. By following some simple guidelines, and asking the correct questions up-front to determine the requirements, even advanced architectures may be straightforward.

Chapter 13: Access Manager for Operating Systems

This chapter describes in detail the Tivoli Access Manager for Operating Systems product. We discuss the architecture and building on the Stocks-4u.com scenarios to introduce its real-world applications.

13.1 Overview

Tivoli Access Manager for Operating Systems provides a layer of authorization policy enforcement in addition to that provided by a native UNIX operating system 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 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.

13.2 MASS perspective

MASS uses the concept of architectural subsystems 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 it to be deployed as a service within a real-world infrastructure.

The main MASS 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 utilizes all MASS subsystems, but these two are fundamental to the Access Manager to MASS mapping and the position 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.

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 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 will be 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 that the applications run on, and the network that supports the solution.
 - Implication: Securing all aspects of an IT system will always generate 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, provide 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 very strong system-level audit capability across a large

environment. This capability, in conjunction with the Access Manager application-level logging, can provide a very comprehensive operational view of the environment.

13.3 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 use 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 as follows.

Core components

Access Manager is based on two components:

- A user registry
- An Authorization Service, consisting of an authorization database and an authorization engine

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, “Introduction to Access Manager components” on page 127](#).

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.

13.3.1 Authorization model

Tivoli Access Manager for Operating Systems components operate in the user-level application space and also 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 13-1 on page 344](#). Applications access system resources through system-provided APIs, which eventually arrive in the UNIX kernel through a variety of mechanisms.

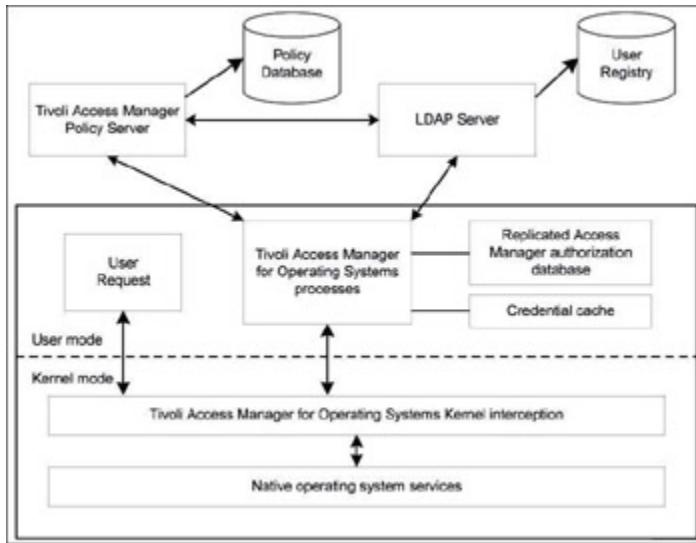


Figure 13-1: An overview of IBM Tivoli Access Manager for Operating Systems

On a system 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 utilizes 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.

13.4 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 (ACL)
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 on 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 what programs can be used to access a resource.

13.4.1 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 attempting the access and the action 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 13-1](#) details the level of access control that can be applied through the file policy.

Table 13-1: 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.
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.

Permission name	Permission granted
Kill (K)	Terminate a process that was executed from a file system resource.
List (l)	List the contents of a directory.

Trusted Computing Base (TCB) resources

Tivoli Access Manager for Operating Systems provides the ability to define files on a system as being part of a Trusted Computing Base. Files that are members of the trusted computing base are monitored for changes in ownership, UNIX file permissions, creation and modification timestamps, 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's signature.

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 Trusted Computing Base 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.

13.4.2 Network policy

Access Manager for Operating Systems provides the ability to control access to remote network services from a local

machine and also to control access to local network services from remote locations. These two types of network access are controlled separately by defining protected resources. The table below describes the network object names that can be used to define a policy.

Table 13-2: 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. This protocol is represented by the string <i>tcp</i> .	A case-sensitive string representing the protocol.

Object name	Description	Type
service	<p>A description of the set of services represented by this resource. For <i>NetIncoming</i> resources, this service represents the service on the local machine to which an incoming connection has been addressed. For <i>NetOutgoing</i> resources, this service represents the service on the remote machine to which a connection attempt is being made.</p>	<p>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.</p>

Object name	Description	Type
host	<p>A description of the set of hosts represented by this resource.</p> <p>For <i>NetIncoming</i> resources, this represents remote hosts from which an incoming connection is attempted. For <i>NetOutgoing</i> resources, this represents the remote host to which an outgoing connection is being attempted.</p>	<p>The host specification may be in one of two forms:</p> <ul style="list-style-type: none"> ▪ ip-address[:nbits] ▪ hostname
ip-address	<p>A dotted notation of an IP address, for example, 192.168.1.42</p>	<p>A string representing an IP version 4 address.</p>

Object name	Description	Type
nbits	<p>The number of bits considered significant in an ip-address. Bits are counted from left to right, 0 indicating that no bits are significant and 32 indicating that all bits are significant.</p> <p>When a host is specified in the ip-address[:nbits] form and no nbits component is specified, 32 is assumed.</p>	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 consisting of wildcard elements and legal host name characters.

13.4.3 Login policy

Access Manager for Operating Systems lets you control when and from where a user can log in to a system. The basic mechanisms for controlling user access are:

- Defining time-of-day login restrictions for users independent of where they log in from
- 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.

13.4.4 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 provided natively by the operating system. The more restrictive between the Tivoli Access Manager for Operating Systems policy and the operating system policy will apply.

13.4.5 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 authorization policy to the User and Group subtypes of the Surrogate resource type. The object names identify the potential targets of surrogate operations and control the ability, for example, to surrogate to the root user or the system group.

13.4.6 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 not only on a command but also on the parameters 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 execute a command as a UNIX user other than that of the invoker.

13.5 Runtime environment

This section describes the major components of Tivoli Access Manager for Operating Systems and their operating environment. The daemons responsible for the major functions of Tivoli Access Manager for Operating Systems are:

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.
pdosIrd	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 timestamp, information identifying the component logging the event, the message classification, and the message text.

13.5.1 The pdosd authorization daemon

The pdosd authorization daemon does the following:

- Handles the authorization requests 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 would 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.

13.5.2 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.

13.5.3 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.

13.5.4 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.

13.5.5 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 will not be running. If login policy is enabled after the initial Tivoli Access Manager for Operating Systems configuration and start, then the pdoslpmd daemon will be started the next time that Tivoli Access Manager for Operating Systems is started.

13.5.6 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.

13.6 Putting it all together

[Figure 13-2](#) on [page 353](#) shows the component interaction associated with a typical process call within an Access Manager for Operating Systems protected system.

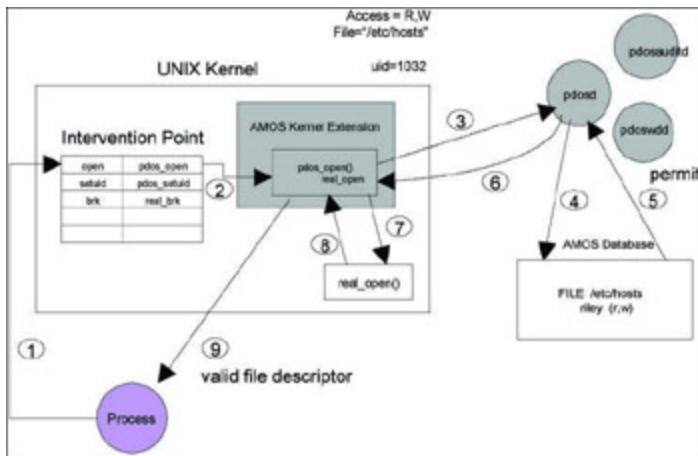


Figure 13-2: Typical Access Manager for Operating Systems component interaction

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 data needed to make an access control decision is:

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 hack 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 will log in under a unique user (non-root) ID. They then **su** 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 off 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.

These are the steps corresponding to the numbers on the diagram:

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 on 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).

3. If the policy is not cached in the kernel then the call is passed up 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. 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 only occurs 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.

13.7 Auditing

Tivoli Access Manager for Operating Systems provides extensive auditing capabilities that permit you to track authorization access decisions made to protected resources as well as to monitor activity of an administrative nature, such as the starting and stopping of the daemons.

Note From a MASS perspective, Access Manager for Operating Systems' core functionality also supports the Audit subsystem. Access Manager for Operating Systems provides strong auditing capabilities at a very granular level. It also supports a consolidated and centralized view of these logs.

13.7.1 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 protected object policy (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 will be audited.

It is possible to enable more granular auditing for authorization access decisions based on the action 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 being performed can be specified separately for global permit, global deny, per-resource permit, and per-resource deny audit levels and 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. Supported audit levels for user-level audit authorization policy are: `permit`, `deny`, `loginpermit`, `logindeny`, `all`, and `none`.

The `permit` and `deny` audit level enables 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 level enables 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. The `none` level 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.

13.7.2 Auditing administrative activity

You can also audit actions taken by the Access Manager for Operating Systems daemons of an administrative nature by setting the `admin` level in the global audit level. The `admin` audit level causes Tivoli Access Manager for Operating Systems daemons to generate audit records for events such

as 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, and the detection of incorrect policy. The admin audit level also causes the generation of audit records for events related to a user login account being enabled or disabled when login activity policy is being enforced. Enabling the info level in the global audit level causes auditing to occur for routine events such as the pdosd daemon receiving valid policy updates. Setting the info level results in a large amount of audit data being generated. The overall space consumed by audit data should be monitored carefully.

13.7.3 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 executed is protected by 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_I global audit level is enabled, and the trace_exec audit level is not enabled, TraceExec audit data is only generated for 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_I 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_I audit levels, instead of trace_exec, will cause TraceExec audit data to be generated only for 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 and when the accessing user's accessor identity is either the root user or the accessor identity and the effective UNIX identity does 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.

13.7.4 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), or 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.

13.8 A business scenario

This section uses the Stocks-4u.com example described in previous chapters. The requirements are based on the requirements defined in [Chapter 7, “A basic WebSEAL scenario” on page 201](#) and [Chapter 12, “Access control in a distributed environment” on page 311](#). The business requirements are mapped to system-level technical security requirements.

13.8.1 Business requirements

In [Chapter 7, “A basic WebSEAL scenario” on page 201](#), the following business drivers were defined for the targeted solution:

1. Provide an enabler for consistent application of security policy across the business. The business cannot afford to create multiple, competing security infrastructures.
2. 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.
3. 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.
4. Allow for the integration of special premium applications capabilities to ML&J’s *select* clients. ML&J is very focused on maintaining its 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.

5. Provide for expansion of services with minimal incremental investment. When the security solution is in place, it must 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.
6. 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.

From [Chapter 12, “Access control in a distributed environment”](#) on [page 311](#), the following business requirements have been proposed by management:

1. Savannah facilities be fully utilized to minimize the San Diego expansion requirement.
2. Internet customer application access must be immune to a San Diego site failure.
3. Management has directed that cross-site telecom costs for internal applications be reduced by 25% and the savings be used to expand network support for customer applications.

13.8.2 Security design objectives

From the above it has been determined that the following key technical system requirements exist:

- Provide a consistent security policy across all Stock-4u.com servers.
- Enable security policy to support the network zones' security classification requirements (that is, security policy that is most secure and locked down in the Internet DMZ, and more open within the more trusted zones).
- Provide a framework to enable the rapid deployment of new server infrastructure.
- Protect application-level transaction down to a data storage level.
- Minimize data flow between the Stocks-4u sites in Savannah and San Diego.
- Provide a consistent audit and logging framework, with centralized log consolidation capabilities for event alerting and auditing.
- Support system administration from multiple locations in a consistent manner.

13.8.3 Requirements analysis

The requirements for this access control subsystem are typical of 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, 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.

13.8.4 Architecture overview

The environment into which Access Manager for Operating Systems is deployed is shown in [Figure 13-3](#) on [page 361](#). 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: Internet DMZ, intranet, management zone, and so on

The environment in [Figure 13-3](#) depicts the San Diego location. The Savannah location has a scaled-down version of this environment. A diagram is not provided, but the main differences are:

- The Savannah management zone only contains systems management servers and the Tivoli Access Manager Web Portal Manager. All other Tivoli Access Manager management components reside in San Diego.
- The Savannah Internet DMZ only contains WebSEAL and DNS servers. The mail servers are located in San Diego.
- Savannah only contains Internet-facing Web and application servers, LDAP replicas, and a Policy Proxy Server.

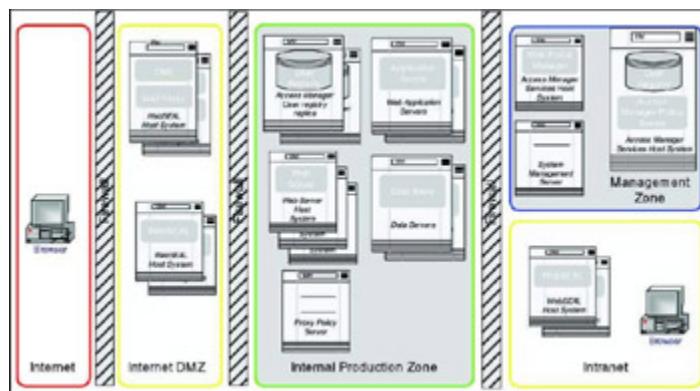


Figure 13-3: Servers deployed within the Stocks-4u Web environment

13.8.5 Policy design

Several options are available within Access Manager for Operating Systems regarding the structure of the security policy for the protected server base. This section describes the key factors that must be considered in the definition of the Access Manager for Operating Systems security policy.

These options relate to the way the ACL policy is defined within Access Manager for Operating Systems. The high-

level structure of the policy has the largest effect on the operational impacts and manageability of the security policy definition. The two major structural elements are:

- Policy branches
- Access Manager for Operating Systems domains

Policy branches are very useful, as they enable security functionality to be centrally grouped and mapped to an infrastructure, hence becoming more effectively managed.

The latest version of Tivoli Access Manager enables the use of multiple Access Manager security domains. This creates an ability to minimize network traffic and system resource impact of ACL database replication within a large environment. Each security domain has its own ACL database, hence an environment with multiple domains will only replicate the security on a per-domain basis, not to the whole environment.

Policy structure

The approach for this scenario is designed to leverage multiple Access Manager domains in conjunction with policy branches within each domain. The main reason for this is to reduce the amount of traffic generated within the overall environment when a policy replication is required.

The domain structure is based on network zones, with policy branches used to group the different types of servers within each zone. Therefore, there will be four Access Manager domains associated with the Access Manager for Operating Systems deployment:

- Internet DMZ
- Production Zone

- Management Zone
- Intranet

These domains will span both physical locations to ensure that the security policy is applied consistently, as depicted in [Figure 13-4 on page 363](#).

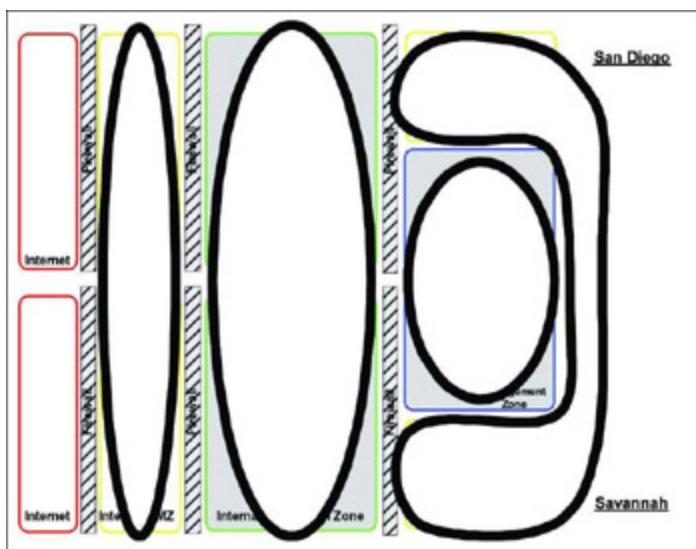


Figure 13-4: Access Manager for Operating Systems domain structure

Note Additional domains will exist for Web resources.

The one user base is shared among all Access Manager for Operating Systems domains to minimize user administration overhead. Individual users and groups can also be defined on a per-domain basis if required; however, for this example this will not be utilized.

Note Another option was to have two domains: one for each location. However this would have a smaller effect on network traffic reductions and could lead to policy discrepancies between the two sites.

Design factors

The main considerations when defining the system security policies are:

- User / administration access; which users will be allowed to manage the servers and under what conditions this will be done.
- Available services for each type of server; that is, what connections a server will accept and what connections are allowed to other servers.
- Logging requirements; addressing what to log and to what level of detail.
- Operational monitoring; snmp alerting, syslogs, process and CPU monitoring, and so on.

User 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 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 required.

- Application managers

This group manages the business applications.

- Security administrators

This group manages user access and user provisioning. They are responsible for creating users and placing them in appropriate groups and so on.

13.8.6 Policy definitions

In this section we describe at a high level the Access Manager for Operating Systems policy deployed within the Stocks-4u environment. We will only describe the policy associated with the Internet DMZ to illustrate what the policies could look like. Applying the design factors to the other Tivoli Access Manager policy domains and servers within them will enable you to derive similar policies to those described below.

Internet DMZ

The Internet DMZ will be defined as one Tivoli Access Manager domain with the following policy branches:

- WebSEAL servers
- Mail and DNS servers

The servers in the DMZ will have the following interfaces:

- External interface
- Management interface
- Internal interface

This zone is probably the least trusted zone within the Stocks-4u Web environment. As a result, the security policy must be as restrictive as possible in order to disable any unnecessary services and user access.

WebSEAL servers

The groups 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 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 Proxy Policy Server. It also accepts SSH connections from the system management server within the management network for CLI administration, and allows outbound connections for systems management traffic only to specific servers, such as TEC and Risk Manager servers.

A Trusted Computing Base is established within the servers to protect core system files such as routing tables, password files, and so on. Alerting on compromise for the Trusted Computing Base is forwarded to Tivoli Risk Manager.

Mail and DNS servers

These servers are solely managed by the network services group, which has full access to the servers.

Inbound connections are allowed to UDP port 53 for DNS and TCP port 25 for SMTP on the external network interface.

Outbound connections are allowed for SMTP to the actual mail servers within the internal network.

The management network allows SSH connections from the systems management servers, and allows the same outbound connections as for the WebSEAL servers.

A Trusted Computing Base is established within the servers to protect core system files such as routing tables, DNS tables, password files, and so on. Alerting on compromise for the Trusted Computing Base is forwarded to Tivoli Risk Manager.

Chapter 14: Access Manager for Business Integration

Overview

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 MASS and the security 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, “Introduction to Access Manager components” on page 127.](#)

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 Queueing 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 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.

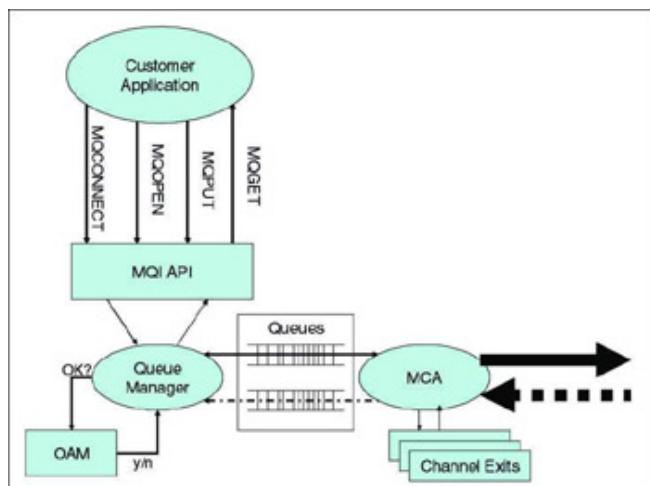


Figure 14-1: IBM WebSphere MQ programming

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.

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 legacy 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](#) on [page 371](#) shows the interaction between various components of WebSphere Business Integration Message Broker. A brief description of the components shown in the diagram follows.

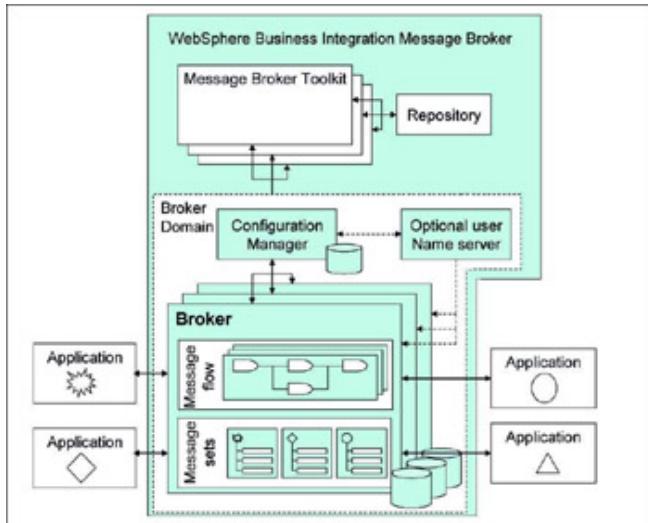


Figure 14-2: WebSphere Business Integration Message Broker overview

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](#).

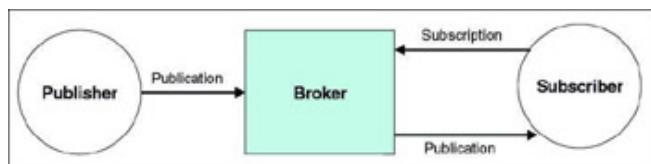


Figure 14-3: Simple publish/subscribe system

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

MASS uses the concept of architectural subsystem to provide a way to group common attributes and to provide a common set of service 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 MASS 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 MASS subsystems. However, the three just listed are fundamental to the Access Manager-to-MASS mapping and the position of 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](#) on [page 380](#) shows a diagram of the core IBM Tivoli Access Manager for Business Integration components and security infrastructure components (in shaded areas).

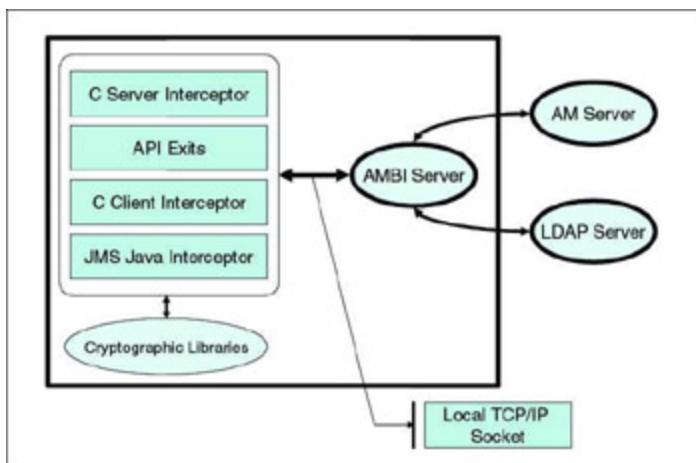


Figure 14-4: IBM Tivoli Access Manager for Business

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, “Introduction to Access Manager components”](#) on [page 127](#).

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](#).

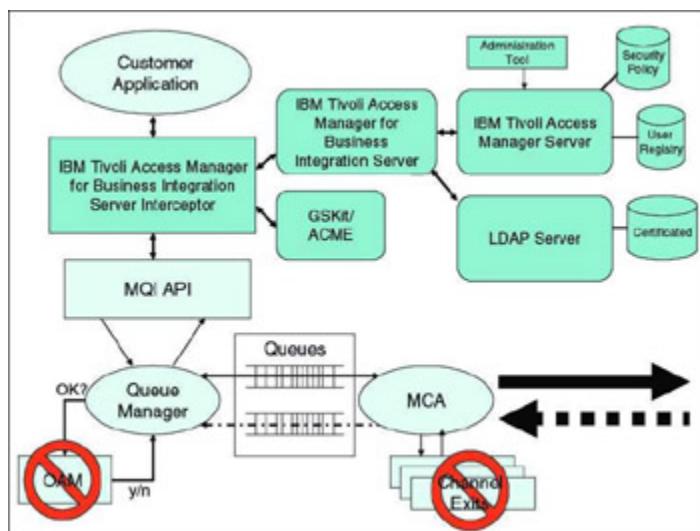


Figure 14-5: Tivoli Access Manager for Business Integration interceptor model

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 on page 385](#).

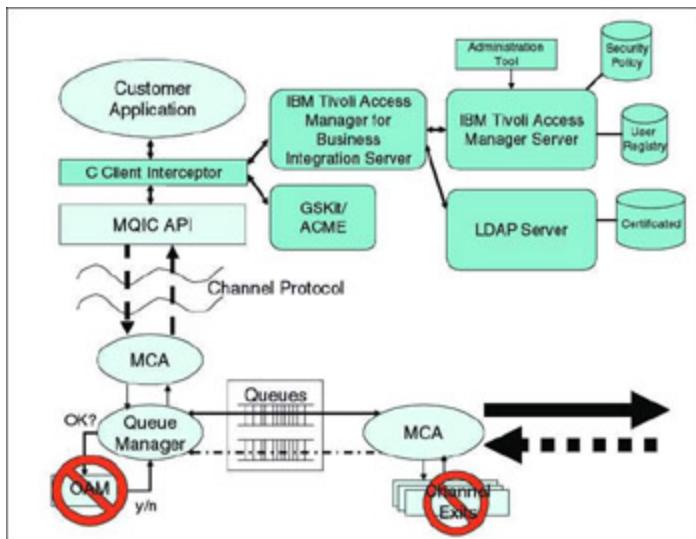


Figure 14-6: IBM Tivoli Access Manager for Business Integration C Client Interceptor

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.

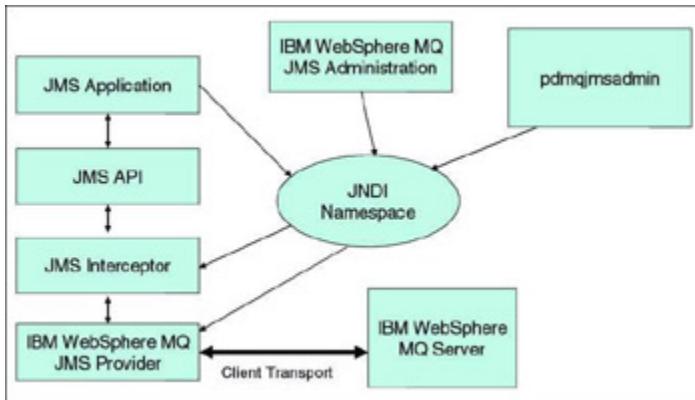


Figure 14-7: IBM WebSphere MQ JMS architecture with JMS Client Interceptor

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.

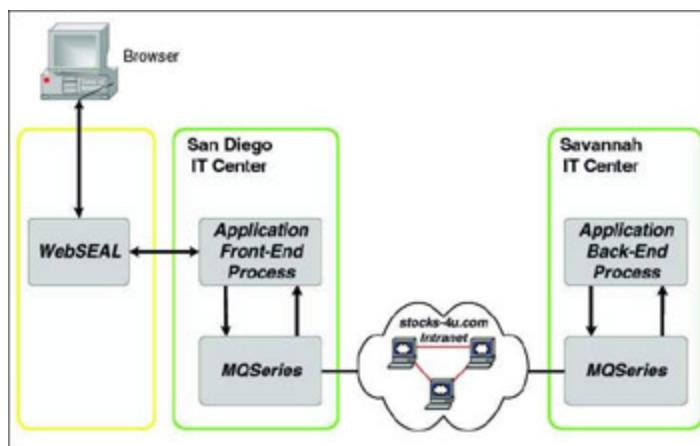


Figure 14-8: A Stocks-4u.com MQSeries application
Savannah

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](#) on [page 390](#), which depicts Access Manager for Business Integration components and interactions.

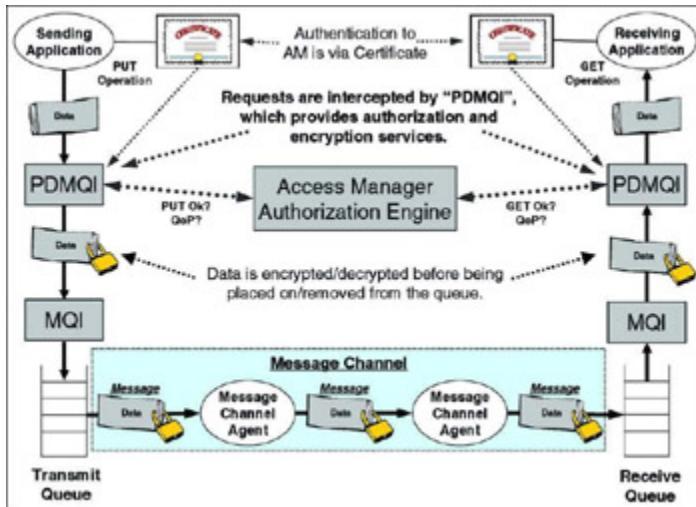


Figure 14-9: Access Manager for Business Integration architecture

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](#).

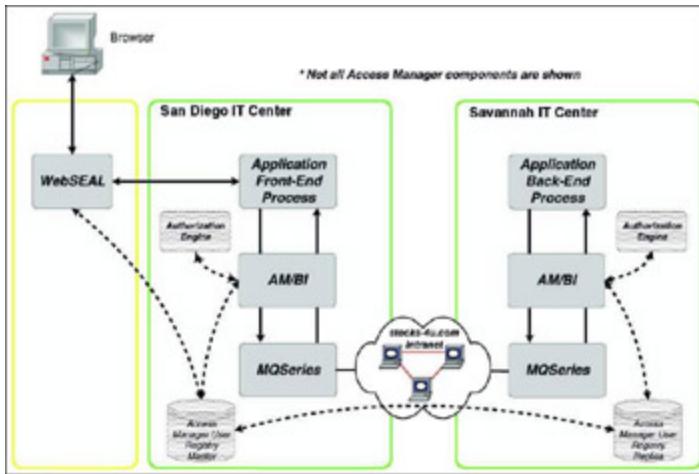


Figure 14-10: A Stocks-4u.com Access Manager for Business Integration scenario

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.

Chapter 15: Tivoli Privacy Manager

This chapter introduces the IBM Tivoli Privacy Manager product. It describes key privacy concepts at a high level and provides the reader with an understanding of the following topics:

- High-level logical component architecture of IBM Tivoli Privacy Manager
- Physical systems in an IBM Tivoli Privacy Manager environment and component placement
- Scalability and high-availability considerations

15.1 Tivoli Privacy Manager overview

IBM Tivoli Privacy Manager for e-business is an enterprise-wide solution for managing access to 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. Using Tivoli Privacy Manager, your organization can identify, monitor, control, and audit access to personal information.

Tivoli Privacy Manager adds the best practices of privacy management to your e-business applications and manages the information these applications collect according to the Organization for Economic Cooperation and Development^[1] (OECD) guidelines on the protection of privacy.

Tivoli Privacy Manager integrates with existing system software, including e-commerce applications and databases. It monitors access to sensitive information and maintains access audit records in separately managed storage.

Tivoli Privacy Manager generates a variety of reports covering both individual and enterprise-wide perspectives. Enterprise-wide reports show the entire collection of privacy-sensitive information and governing policies, and provide audit trails showing accesses on monitored systems that indicate whether the accesses conform to governing policies. Individual reports show audit trail information about who accessed an individual's personally identifiable information (PII) and for what purposes.

Tivoli Privacy Manager reports enable the CPO to determine:

- Where privacy-sensitive information is stored within the organization
- The privacy policy and its elements that govern privacy-sensitive information
- Which information-access requests are in conflict with the governing policy
- Who has accessed or attempted to access an individual's privacy-sensitive information within a specified period

15.1.1 Privacy-sensitive information

Privacy-sensitive information is data for which access and usage is monitored according to OECD principles. According to these principles:

- There must be no personal data record-keeping systems whose existence are secret.
- There must be a way for an individual to determine what personal information is in a record and how the information is used.
- There must be a way for an individual to prevent personal information that was obtained for one purpose from being used or made available for other purposes without the individual's consent.
- There must be a way for an individual to correct or amend a record of identification about the individual.
- Any organization creating, maintaining, using, or disseminating records of personal identification must ensure the reliability of the information for its

intended use and must take precautions to prevent misuse of the information.

OECD principles provide the regulatory framework within which privacy can be managed:

- Collection limitation

Only the information needed for transactions is collected.

- Information quality

Collected personal information should be relevant to the purposes for which it is used, and should be accurate, complete, and up-to-date.

- Purpose specification

The purposes for which personal information is collected should be specified at the time the information is collected and subsequent use should be limited to the fulfillment of specified purposes.

- Use limitation

Personal information should not be disclosed, made available, or otherwise used except with the consent of the information owner or by the authority of law.

- Notification and consent

Individuals should be notified and required to provide consent for each purpose and use of personal information.

- Openness

A means should be readily available for determining the existence and nature of personal information, and the main purposes of its use, as well as the identity and usual residence of the information controller.

15.1.2 Basing a privacy policy on P3P

Tivoli Privacy Manager uses the Platform for Privacy Preferences (P3P) model to create policy definitions. The P3P specification was developed by a World Wide Web Consortium (W3C) working group. W3C develops interoperable technologies (specifications, guidelines, software, and tools) for the Web. The P3P specification is a privacy-protection technology that provides the means to generate policies that can be posted, accessed, and evaluated for privacy compliance in a machine-readable format (XML). A growing list of major Web sites have adopted P3P as the means to convey privacy policies to users and to provide a technology by which Web users can determine whether a Web site's privacy policy meets their privacy requirements. Refer to the IBM Tivoli Privacy Manager Planning Guide for more information about how Tivoli Privacy Manager uses the P3P specification.

15.1.3 How Tivoli Privacy Manager works

Privacy management includes the following key tasks:

1. Define the privacy policy.
2. Deploy the policy to the IT systems that contain privacy-sensitive information.
3. Record consent of users to advertised privacy policy when they submit privacy-sensitive information.

4. Create an audit trail of access to privacy-sensitive information.
5. Generate enterprise-wide and individualized reports showing accesses to privacy-sensitive information and conformance to the governing privacy policy.

Defining a privacy policy

Privacy policies are defined by people who understand the business and legal environment of an organization. These individuals can express the conceptual requirements of applicable privacy law and business strategy. Typically, privacy policies do not refer to specific technologies, applications, or systems in an IT infrastructure. A privacy policy:

- Defines what types of information must be managed as privacy-sensitive by an organization.
- Stipulates how privacy-sensitive information can be used.
- Regulates who can access and use privacy-sensitive information within an organization.

Deploying a privacy policy

After the CPO staff publishes the organization's privacy policy, the IT staff performs the tasks that enable the policy to be deployed into the organization's network.

The first main task identifies where privacy-sensitive information is stored in the organization and classifies these storage locations with one or more PII types from the privacy policy. A storage location is the location of a data instance in a monitored system.

The second main task in deploying a privacy policy is similar to the first task: It associates groups defined in the policy with authorized groups in the organization's security infrastructure.

Recording consent to a privacy policy

After a policy is deployed into the IT infrastructure by the IT staff, Tivoli Privacy Manager monitors begin to watch for information flowing in and out of monitored systems. One of the key tasks of a Tivoli Privacy Manager monitor is to detect the submission of new or updated information in the privacy-sensitive fields of the monitored system.

When information is submitted or updated, Tivoli Privacy Manager records the identity of the individual whose information is being recorded, the associated PII type, and the time of the submission, as a submission record. From this information, Tivoli Privacy Manager determines the privacy policy that the submitter consented to and, therefore, the policy that will govern all future use of the information.

Monitoring and enforcing information access

In addition to watching for the submission of information from information owners, a Tivoli Privacy Manager monitor watches for applications attempting to access privacy-sensitive fields in a monitored system. When an access attempt occurs, the monitor collects enough information to create an audit trail of the access. The monitor identifies whose information is being accessed, its PII type, who is accessing the information, the intended purpose of the access, and the time that the access occurred. It sends this information to the Tivoli Privacy Manager server. The server then locates the submission record corresponding to the

information that was accessed and the governing privacy policy statement. Using the governing statement, the Tivoli Privacy Manager server makes a conformance check to determine whether the access conformed to the governing privacy policy.

Generating reports

The Tivoli Privacy Manager server uses information supplied by monitors to create and store audit trail records in a database. These audit records are used in conjunction with settings from the Tivoli Privacy Manager console to generate various types of reports, including reports about monitor settings, policy settings, and specific histories of submission and access activity on privacy-sensitive information.

Report criteria are specified to produce and save report definitions for the various reports. Report definitions are then used to generate reports on demand, according to the needs and requirements of the organization.

15.1.4 Privacy vs. security

Ensuring security in an organization is the foundation for protecting the privacy of individuals. Privacy can be compromised by a failure in security; however, privacy addresses the use and misuse of information by authorized users. Privacy is an information-handling policy rather than an access-control policy. Together, privacy and security form the foundation for establishing trust.

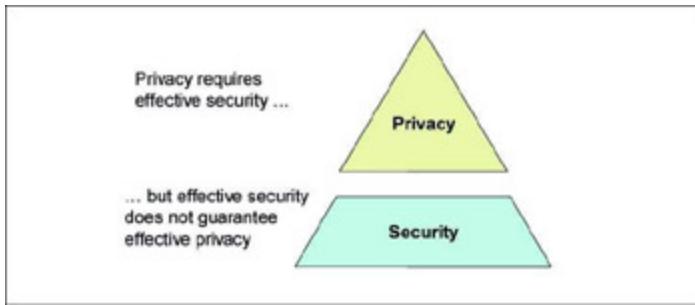


Figure 15-1: Solid security is the foundation of privacy protection

From an overall MASS perspective, Tivoli Privacy Manager sits within the access control and audit subsystems. However, this is misleading, as the access control functions provided by Tivoli Privacy Manager actually are delivered with Tivoli Access Manager. Therefore, architecturally speaking, Tivoli Privacy Manager sits within the audit subsystem. It does not in itself enforce access control policy; it only monitors and reports on authorized access to privacy sensitive information.

[1]The OECD Web site provides different articles on the background of privacy guidelines: <http://www.oecd.org>

15.2 Tivoli Privacy Manager architecture

Deploying enterprise software solutions such as IBM Tivoli Privacy Manager usually have a high impact on the overall IT architecture of a company. IBM Tivoli Privacy Manager consists of numerous components that must be placed onto physical systems. A good understanding of the components, what they do, and how they communicate with each other is key to a good architecture design.

Note The IBM Tivoli product name is IBM Tivoli Privacy Manager, but we use the term *Privacy Manager* when we consider the product with all of its components.

The main components of IBM Tivoli Privacy Manager are monitor, server, and database. Our environment contains several servers and databases so we instead use the terms *monitors*, *privacy server*, and *privacy database* respectively. A privacy-enabled application consists of the application itself and a place where the application stores its data, so we call the components *application* and *data store*.

15.2.1 Basic application components

When we talk about applications, we consider Web applications as client-server applications. The browser is the client component and the Web server, Web application server, and database server are the server components.

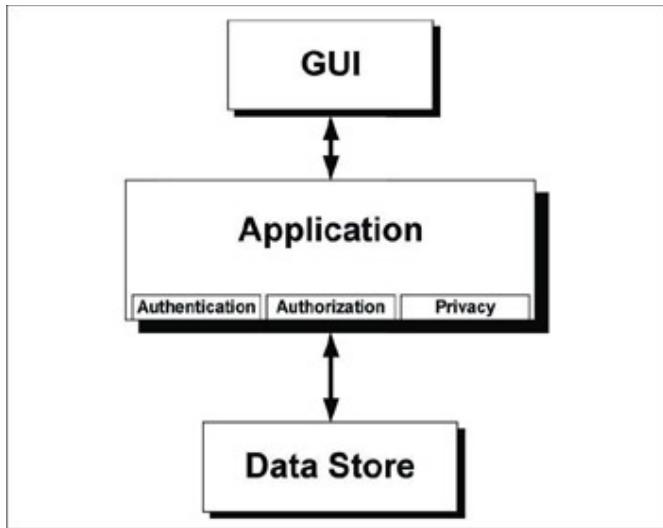


Figure 15-2: Basic components of a client-server application

As shown in [Figure 15-2](#), the application components of a client-server application consist of a graphical user interface (GUI), an application, and a data store. The application contains hard-coded functions to enforce authentication, authorization, and privacy. A user has to log on to this particular application where it is determined, if he is allowed to use the application, what functions he can execute and what data he can see and manipulate. In all applications of a company, these functions have to be addressed by the application itself. Concerning privacy, the rules are hard-coded within the application and enforced by the business process. For example, before a list of customer purchases was given to the Marketing department for market research purposes, the individual customer identifying data had to be masked as unreadable.

15.2.2 Centralized privacy service

Because privacy is a concern in many applications it makes sense to isolate the privacy service in a similar way to that of the authentication and authorization services. This follows the architectural approach as adopted by Tivoli Access Manager (that is, it externalizes the application privacy services).

Figure 15-3 shows how the privacy service is centralized and isolated for the application.

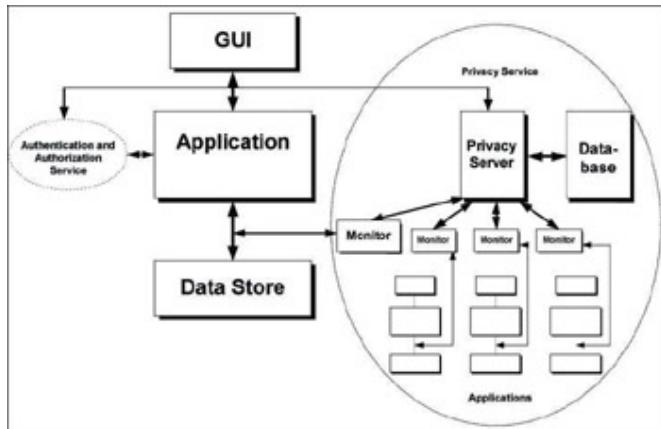


Figure 15-3: Centralized privacy service

As shown, the privacy service is isolated from the application. It is provided as a service to all applications in a company or in a domain.

To implement a privacy service, the application (which must be privacy-enabled) has to be intercepted in order to check the privacy rules before data can be presented to a user. The interception is implemented with a monitor. If the application accesses personally identifiable information (PII) the monitor collects additional information and passes this information on to the privacy server. The privacy server makes a decision, whether all, some or none of the data elements the user wants to access can be exposed. To enable the centralized privacy enforcement, the privacy server stores the privacy rules and consent information in its data store. Consent defines, what needs to be wiped out, when data is accessed for a certain purpose. Consent can be given or denied by the customer or by company internal regulations and policies.

In the IBM Tivoli product world, the privacy services are provided by the IBM Tivoli Privacy Manager, which itself uses a DB2 database as the data store. Monitors are enforcing data access restrictions based on privacy policies.

Isolating a service moves the complexity from a single application into the IT infrastructure. The principle of *economy of scale* is applied to computer applications. By allowing faster development, maintenance, and deployment of applications it saves costs and reduces security risks. As another important factor this particular IT architecture enables an enterprise to get prepared for the *e-business on demand* age, because centralized services can be relocated to any place on earth or even be out sourced.

15.2.3 Layered architecture for privacy management

In this section we describe a layered architecture for privacy management. [Figure 15-4 on page 402](#) depicts the three layers:

- User interface layer
- Application layer
- Service layer

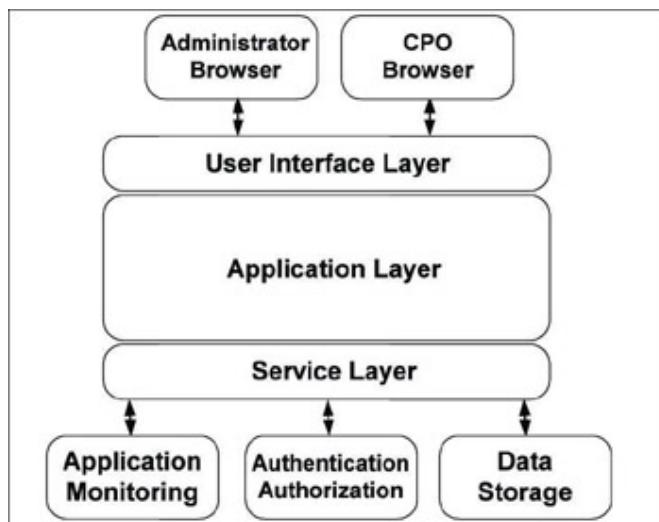


Figure 15-4: Layered architecture for privacy management

The user interface layer contains the graphical user interfaces for privacy management administrators and for the Chief Privacy Officer and his staff. These interfaces are Web-based and can be accessed with a Web browser.

The application layer is underneath the user interface layer. This is where privacy policies and monitor data are managed, reporting functions are executed, and user consent is updated. The monitors trigger procedures in the application layer to make the decision about which data fields can be exposed to the application end user. The application layer uses the features provided by the service layer.

The service layer supports the application layer by providing basic functions such as data storage and an authentication and authorization service. Data storage is provided through a relational database where privacy policies, monitor data, consent information, and audit reports are stored. The authentication and authorization service answers such basic questions as whether a user is who he claims to be and what functions the user is allowed to access. The most important feature of the service layer is the application monitoring service. Application monitoring is the interception of applications to ensure that data access is conforming to privacy policies.

15.2.4 Component description

In this section we take a closer look at the IBM Tivoli privacy management solution implemented with IBM Tivoli Privacy Manager and describe in detail the components of the privacy management environment shown in [Figure 15-5](#).

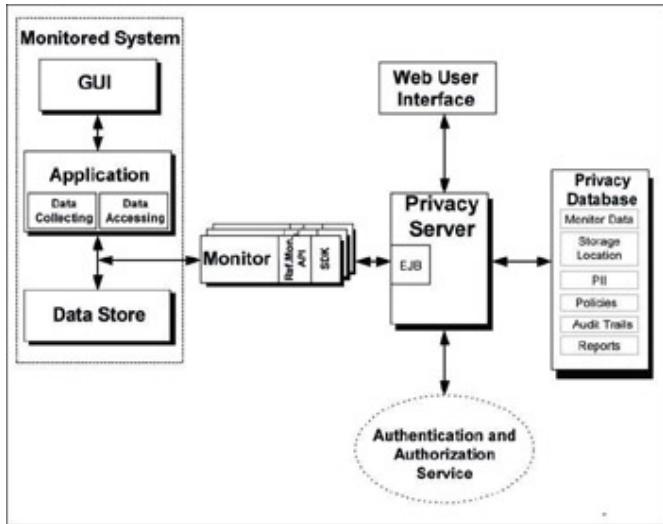


Figure 15-5: Components of a privacy management solution with IBM Tivoli products

Monitored system

The *Monitored System* is the customer application that must be privacy enabled. Users of a monitored system collect PII or access PII, which is stored in the *Monitored Systems Data Store*. The Monitored System is accessed through a GUI. Examples for Monitored Systems are ERP applications, CRM applications, and directories.

Web user interface

The *Web user interface* connects to the *Privacy Server*, the actual Privacy Manager engine that runs on the IBM WebSphere Application Server. The Web user interface, named *Privacy Manager Console*, is implemented as a combination of Java Server Pages, Servlets, and Enterprise Java Beans. It enables its users to accomplish the following tasks:

- Create and modify, export, and import privacy policies.
- Administer monitors.
- Create and manage report definitions.
- Start and manage reports.

Users can be grouped into the following categories:

CPO staff	Performs the tasks associated with creating and publishing a privacy policy as well as the tasks that an auditor can perform. (See Auditor below.)
IT staff	Performs the tasks required to set up privacy policies and monitors for deployment and to generate audit reports.
Administrator	Designed to perform a small number of tasks, such as deleting persistently stored records that are no longer needed and configuring the authorization service to enable Tivoli Privacy Manager to obtain and publish the users and groups defined to that service.
Auditor	Designed as an external auditing role to create, manage, and run audit reports that provide a historical accounting of PII submission and access in the monitored system. These tasks can also be performed by the CPO staff and IT staff roles.

Privacy Server

The Privacy Manager *Privacy Server* is the main processing component in a privacy management architecture. It processes incoming requests from the Web user interface and from monitors. The Privacy Server is connected to one or a set of monitors. Incoming requests from monitors ask for decisions, whether data elements can be exposed to users of an application, or new consent submissions. In case of data access, a monitor request contains details about the identity of the user, which storage locations are accessed, the purpose of the access, and other defined conditions. After the Privacy Server has decided, a response is sent to the monitor.

The decision is made with consideration of data stored in the Data Store and by the use of an authentication and authorization service. Enquiries from the Web user interface are requests to create or modify Privacy Manager configuration data such as privacy policies, monitor definitions, and reports. This data is also stored in the Data Store. The Privacy Server component is an Enterprise Application deployed on IBM WebSphere Application Server Advanced Edition.

Privacy Database

The *Privacy Database* contains tables for the following data:

- P3P (Platform for Privacy Preferences) policies Identity and group mappings
- Usage purposes
- Conditions
- Report definitions
- Data submissions
- Data access operations

These database tables are used by the Privacy Server when privacy decisions are made and when reports are to be generated. The Privacy Database is implemented using the IBM DB2 Universal Database.

Monitors

To privacy-enable an application, one or more *monitors* must be developed. Monitors are adapters that exchange information with the application and the Privacy Server, and are the policy enforcement point within the architecture. Monitors are implemented using the Monitor SDK to communicate with the Privacy Server. They also can be developed using the Reference Monitor API, which simplifies the development. (See “[Reference Monitor API](#)” on [page 406](#).) Monitors have the following functions:

- Poll the Privacy Server to find out whether any new configuration data, such as policy changes or monitor definition changes, is important to the monitor.
- Intercept an application to check whether data access to PII conforms to a defined privacy policy.
- Pass new consent submissions to the Privacy Server.
- Inform the Privacy Server that PII data is accessed. This is performed for the purpose of reporting.

Monitors are deployed in a WebSphere container on a WebSphere Client or WebSphere Server.

SDK

The *Monitor Software Development Kit* (SDK) is a tool that enables customers to develop customized monitors. It contains Java APIs that are used by a monitor to connect to the Privacy Server and request enforcement decisions. The SDK is of primary interest because in most customer-specific deployment

cases monitors need to intercept existing applications that must be privacy-enabled. The SDK provides the following functions:

Register a monitor and get information about a registered monitor.

- Register and manage storage location lists.
- Submit PII storage locations and report of PII data access.
- Check conformance of real-time privacy policy.
- Handle monitor exceptions.

The SDK is located in a WebSphere container on a WebSphere Server or WebSphere Client. The Client uses the Internet Inter-ORB Protocol (IIOP) to talk to EJBs on the Privacy Server.

Reference Monitor API

The *Reference Monitor* for Tivoli Privacy Manager is a reference implementation of a Tivoli Privacy Manager monitor, consisting of sample code that can be modified quickly to build privacy functionality into new or existing applications. The Reference Monitor's purpose is to simplify the task of developing new privacy monitors. The Reference Monitor API makes use of the Monitor SDK. Typically most new monitor developments use the Reference Monitor API, because it enables rapid monitor development.

Reference Monitor provides a low-level foundation API and a sample Reference Monitor, which also includes the complete Java source code of its implementation. The code is necessary for customizing a reference monitor for a new application. The Reference Monitor is not part of the IBM Tivoli Privacy Manager product, but it can be downloaded from IBM alphaWorks® at <http://www.alphaworks.ibm.com>.

Note As this book is published there are two monitor implementations available on alphaWorks: the

Reference Monitor as described above and a monitor for HIPAA that adds privacy controls to the *IBM WebSphere Business Integration Collaboration for HIPAA Transaction* by monitoring claims transactions between healthcare payers and providers, ensuring that privacy policies and individual privacy choices are respected.

- The Reference Monitor for Tivoli Privacy Manager can be found at
<http://www.alphaworks.ibm.com/tech/refmon>
- The Privacy Manager Monitor for HIPAA can be found at
<http://www.alphaworks.ibm.com/tech/hipaa>

Authentication and authorization service

The Privacy Server uses Tivoli Access Manager to associate authorized users and groups with the groups defined in a privacy policy. The Privacy Server maps application-specific users and groups to Tivoli Access Manager.

15.3 Physical component architecture

In this section we describe how the logical components can be mapped onto a physical infrastructure. A privacy IT architect has many options to choose among for component placement, so we demonstrate an architecture that uses a dedicated server node whenever possible, as well as the functions of each server node and list the advantages for this configuration. In [15.3.2, “Enterprise Privacy Server configuration”](#) on [page 412](#), we describe an architecture that centralizes as many components as possible. These two sample architectures provide the reader with a good overview of possible solution architectures.

15.3.1 Dedicated server configuration

[Figure 15-6](#) illustrates the dedicated server configuration.

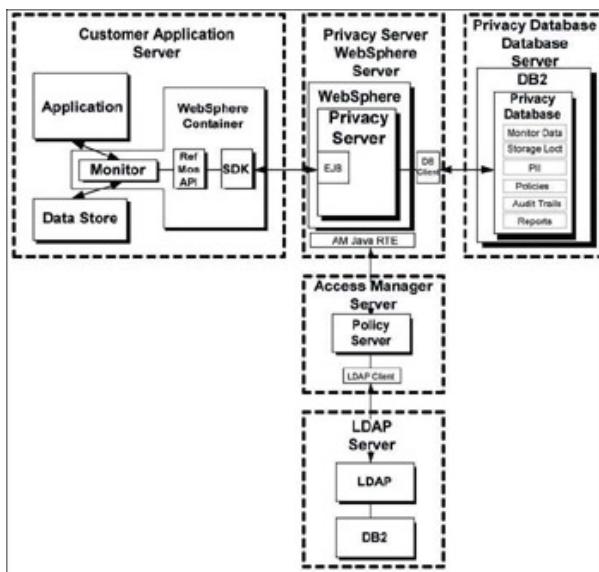


Figure 15-6: Dedicated server configuration

The following sections describe all major components of the dedicated server configuration.

Customer application server

The *customer application server* contains the application and the application data store. A Privacy Manager monitor must be installed to privacy-enable a customer application. The Privacy Manager monitor

itself requires the Monitor SDK and a WebSphere container as a runtime environment. In the following each component is described.

Application

These are customer-specific applications that must be privacy-enabled. Usually these applications cannot be changed and have to be taken as a given. Applications can be of any type, such as Web-based, client-server, or mainframe-based. Therefore they can be located on any type of server in the customer environment. For interception by a Tivoli privacy monitor, ideally the application is a Java application with well-documented access routines to its data store. A WebSphere client has to be available on the application server as well to provide the run-time environment for the monitor. If the customer application is not Java based, other ways must be explored to intercept the application.

Application data store

Usually this component is also a given and determined by the user environment. The application data store can be a relational database, a file or file system, a proprietary database, or an LDAP user registry. It can be located on a Windows or UNIX server or on a mainframe. Whatever mechanism the customer application uses to access the data store, it has to be intercepted by the monitor. Often the application data store is physically co-located with the application.

Monitor

This component has to be developed to intercept the application and the application data store. It requires the Monitor SDK. Whenever possible, a new monitor development should be based on the Reference Monitor.

Monitor SDK

The Monitor SDK is the Java API that is used to develop monitors. The Monitor SDK has been installed in a WebSphere container.

WebSphere container

The WebSphere container is required as a runtime environment for the monitor and can be provided by either a WebSphere Client or a

WebSphere Server. In this WebSphere container the monitor and the Monitor SDK are installed.

The following products must be installed on the customer application server:

- Customer application
- Customer application data store
- WebSphere container
- Monitor
- Monitor SDK

Privacy Server WebSphere Server

The *Privacy Server* is a Java application that runs on IBM WebSphere Application Server Advanced Edition. WebSphere Application Server is a J2EE-certified application server that provides the runtime environment for the Privacy Server. The Privacy Server runs in one Java Virtual Machine (JVM) and uses a Web container and an Enterprise Java Bean (EJB) container. The Web container is used for the Web user interface, and the EJB container is used for the Privacy Server core functions. To access the *Privacy Database*, a DB2 client must be installed on this server. The Access Manager Java Runtime Environment is used to provide access to the Tivoli Access Manager Policy Server to resolve user-to-group mapping. The following products must be installed on the Privacy Server WebSphere Server:

- WebSphere Application Server
- DB2 client
- Access Manager Java Runtime Environment
- Privacy Manager Server

Note The difference between the Access Manager Runtime Environment and the Access Manager *Java* Runtime Environment is that the first one contains libraries and supporting files that all Access Manager components share as a basic library. The Java Runtime Environment

is used by any Java application that wants to use the Access Manager authorization services.

Privacy Database Server

The Privacy Database Server maintains all the data that the Privacy Server stores. It contains one database with database tables for monitor data, storage locations and other details of Personal Identifiable Information (PII), privacy policies, and audit reports.

The following products must be installed on the Privacy Database Server:

- DB2 Universal Database
- Privacy Manager database

Access Manager Server

Tivoli Privacy Manager uses Tivoli Access Manager as the user and group registry. Tivoli Access Manager is used to check whether the user who is accessing data belongs to a certain group. The privacy policies are defined to enable certain groups to see certain data elements.

The following components must be installed on the Access Manager Server:

- Access Manager Runtime Environment
- Access Manager Policy Server

Note Privacy Manager utilizes the pdadmin APIs in order to communicate with Access Manager. Therefore the Access Manager Policy Server must be available at all times to answer Privacy Manager requests because it is the only Access Manager component that accepts pdadmin API calls.

LDAP Server

The *LDAP Server* is a requirement for Tivoli Access Manager, which stores user and group definitions as well as metadata in the LDAP server. The following products must be installed on the LDAP Server:

- IBM Tivoli Directory Server
- DB2 Database

We note a special role that the LDAP Server can play, apart from its role as data store for Tivoli Access Manager. Because an LDAP Server is a common data store for PII data, it is practical to monitor it for privacy reasons. In this case, an LDAP Server is nothing other than a data store for a customer application. Because the LDAP Server is a common place to store PII data, and an LDAP Server provides a well-defined interface, Tivoli Privacy Manager comes with a set of monitors for LDAP Servers.

Network connections

[Figure 15-7 on page 411](#) uses the dedicated server configuration to illustrate the network traffic and the protocols used between the components.

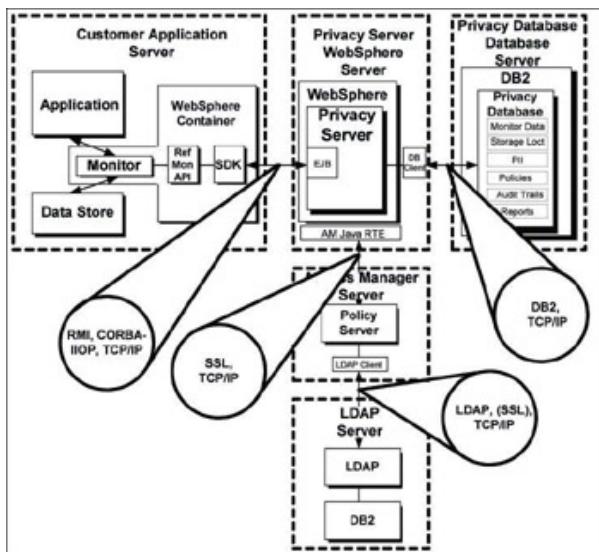


Figure 15-7: Network Protocols between components of Dedicated Server Configuration

Because Monitor and Privacy Server are EJB applications, they use Java Remote Method Invocations (RMI) over CORBA-IIOP on top of TCP/IP to communicate with each other.

Note In Tivoli Privacy Manager Version 1.2, the Privacy Server implements a Web Services interface based on SOAP for

Monitor to Privacy Server communication.

The Access Manager Java Runtime Environment uses a secure protocol based on SSL to communicate to the Access Manager Policy Server.

The Access Manager Policy Server uses an LDAP client that communicates with the LDAP server using the LDAP protocol, which is based on TCP/IP. Optionally, SSL can be switched on between LDAP client and server to encrypt the data packets. The DB2 client uses a DB2 protocol on top of TCP/IP to communicate with the DB2 server.

Usually all components of the privacy management solution are located in the secure zone behind all firewalls. However, consider a case in which the customer application's front-end is located in the Demilitarized Zone (DMZ) and the data store is in the secure zone. Because of monitor design issues, the monitor could now be located in the DMZ, close to the application front-end or in the secure zone close to the data store. In the case of the monitor being located in the DMZ, the monitor communication would have to cross a firewall. This would require that CORBA-IIOP must run across the firewall.

Advantages of dedicated server configuration

The dedicated server configuration has the following advantages:

- Using individual servers for each function increases deployment flexibility.
- Tivoli Access Manager is not tied to Privacy Manager, so it can be used as the centralized Authentication and Authorization Service for other applications as well.
- If one server has a failure, the other servers still work and provide their functionality.

15.3.2 Enterprise Privacy Server configuration

Figure 15-8 shows the Enterprise Privacy Server configuration.

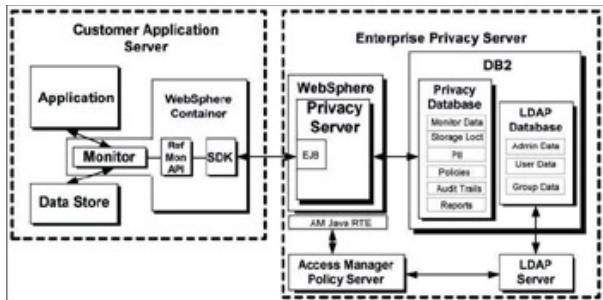


Figure 15-8: Enterprise Privacy Server configuration

In the Enterprise Privacy Server configuration, components are concentrated on one server as much as possible. This means that we have one server that runs the customer application with its data store and the monitor, which requires a WebSphere container and the Monitor SDK. All other components are installed on the Enterprise Privacy Server. This server contains the IBM WebSphere Application Server with the Privacy Server EJB application, and the Access Manager Policy Server components with its Java Runtime Environment. Also located on this server are the LDAP server and the DB2 database, which contains the Privacy database and the LDAP database.

Advantages of Enterprise Privacy Server configuration

The Enterprise Privacy Server configuration has the following advantages:

- Fewer servers means less server management effort, which means lower operational costs.
- Components can communicate faster because they use local server-internal channels.
- There are fewer components. (For example, a separate DB2 client is not needed.)

15.3.3 Other parameters that influence the architecture

As can be seen from the three configuration examples shown, there are many ways the logical components of a privacy management solution can be mapped onto physical systems. These parameters also should be considered when designing a privacy management solution:

- The WebSphere Application Server itself consists of several components, which can be separated onto different machines.
 - The DB2 database has many files, which can be stored on different hard disks.
 - Firewalls may influence architectural design.
 - The Monitor design might require that certain components be installed on a certain server.
 - Scalability, performance, and high availability considerations might influence architectural design.
-

15.4 Scalability, high availability, and performance

The following sections describe considerations, when designing a Privacy Management solution, regarding scalability, high availability, and performance.

15.4.1 Scalability considerations

Scalability is the ability of an application to be used more frequently or by more users without the loss of performance. A typical scenario around a centralized privacy service starts with one privacy-enabled application. Later, other applications are migrated to use the central privacy service. Another scenario involves an application that is used more frequently because of increasing demands, which means that more privacy-related checks must be performed.

A good architectural design considers and provides scalability. Scalability is closely related to high availability in that application environments must have a backup system for every component. This implicitly means that the application is also scalable because a second component can be activated in the case of high load instead of only failure of a component. For this reason, we do not explicitly describe scalability for each component, but we do for high availability.

15.4.2 High availability considerations

When considering *high availability* we can look at the physical components as they are described in 15.3, “Physical component architecture” on [page 407](#). A closer

examination shows that we need to consider the following components for high availability:

- LDAP Server
- DB2 database for Privacy Server
- Tivoli Access Manager
- Privacy Server
- Monitors
- Customer's privacy-enabled application

In the following subsections we give hints or references to help implement high availability of only the Tivoli Privacy Manager related components. High availability and scalability of the LDAP Servers is addressed in 4.2.9, “Availability and scalability” on [page 100](#), and of Tivoli Access Manager components in [Chapter 8, “Increasing availability and scalability”](#) on [page 215](#).

DB2 database for Privacy Servers

Because this book does not focus on DB2 details, refer to the redbook *DB2 Warehouse Management: High Availability and Problem Determination Guide*, SG24-6544. There you can find information about setting up a DB2 database for high availability.

Privacy Server

Because Privacy Server is a J2EE application running on an IBM WebSphere Application Server, a clone application server must be configured on a different machine. Our example calls the two machines *original* and *clone*.

On the clone server we have to install a DB2 client, and the Privacy Manager and WebSphere Application Server databases must be cataloged for this DB2 client. The properties file has to be copied from the original server to the clone server. Both nodes, original and clone, should be displayed when running the WebSphere Administrative Console on both nodes. Using the WebSphere Administrative Console on the clone node, the Application Server clone must be created in the PM group. Privacy Manager and the Access Manager Java Runtime Environment have to be installed and configured on the clone server. The privacy.ear file must be copied from the original server and expanded on the clone.

Monitors

Backup monitors are only needed if the monitors are separated from the application. Most likely, the monitors are tightly connected to the applications and thus can be considered part of the application. In the case where the monitors are separated from the application, a backup WebSphere container with the installed Monitor SDK and the monitors have be triggered to take over and interface with the application. The backup monitors have to be registered with the Privacy Server.

Privacy-enabled application

If the privacy-enabled applications require high availability, so must the monitors. In case of a switch-over, the monitor must re-register with the Privacy Server and reload the configuration.

15.5 Conclusion

Privacy legislation for the private sector is in effect today and most organizations have a published privacy policy. However, most also fail to live up to their policies by lacking the appropriate processes, internal policies, and tools that enable the integration of privacy into their business processes.

Having robust security does not ensure that privacy policies are being supported. Tivoli Privacy Manager provides the tools to apply a corporate privacy policy to a company's privacy-sensitive information. This is achieved by being able to monitor and report, at a very granular level, access to privacy-sensitive information.

The Tivoli Privacy Manager architecture is based on open standards and supports a services-based architecture, which mean that it can be integrated easily with most Internet technology-based applications. Additionally, Tivoli Privacy Manager leverages an Access Manager infrastructure for its policy enforcement capabilities.

More detailed information and best practices for Tivoli Privacy Manager can be found in the IBM Redbook *IBM Tivoli Privacy Manager Solution Design and Best Practices*, SG24-6999.

Part 3: Managing Identities and Credentials

Chapter List

[Chapter 16:](#) Identity Management

[Chapter 17:](#) Introducing Directory Integrator

[Chapter 18:](#) Identity Manager structure and components

[Chapter 19:](#) Identity Manager scenarios

[Chapter 20:](#) Synchronizing the enterprise

In this part, 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.

Chapter 16: Identity Management

Previous sections of the book have dealt with session management. This section deals with another critical area, user lifecycle management, or identity 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. This chapter attempts to outline methods of identifying the key components of an identity management architecture using IBM Tivoli Identity Manager.

16.1 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 life cycle 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 four key areas:

- Identity lifecycle management (user self-care, enrollment, and provisioning)
- Identity control (access and privacy control, single sign-on, and auditing)
- Identity federation (sharing user authentication and attribute information among trusted Web services applications)
- Identity foundation (directory, directory integration, and workflow)

As the world of e-business 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 superset 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.

16.2 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.

16.3 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 book. 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 with in 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 choose 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.

16.4 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 16-1](#), the base of which is the most core required capability of the provisioning solution. After the capabilities in the lowest level have been 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 16-1: The eight levels of identity management

16.4.1 Connectors to access controlled systems

In order to automate provisioning, the solution must communicate securely with each target system being managed. If the connector does not exist, then an administrator must still make the required changes manually. The connectors, or agents, 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 connector 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
- Meta directories

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 Windows NT, AIX, and Linux
- SecureID
- Tivoli Access Manager
- Network devices
- RACF

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.

16.4.2 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

16.4.3 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 and offer to suspend, roll back, or accept the 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 on a scheduled bulk 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

16.4.4 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

16.4.5 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

16.4.6 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 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

16.4.7 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

16.4.8 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
-

16.5 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 using the IBM Method for Architecting Secure Solutions (MASS) can be found in [Chapter 2, “Method for Architecting Secure Solutions”](#) on [page 15](#). More about business process management can be obtained from the redbook *Continuous Business Process Management with HOLOSOFX BPM Suite and IBM MQSeries Workflow*, SG24-6590.

16.6 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 such as MASS.

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 standards.

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 16-2](#) on [page 434](#).



Figure 16-2: 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 16-3 on page 435](#) shows a set of generic steps or phases that relate to the architecture and design document.

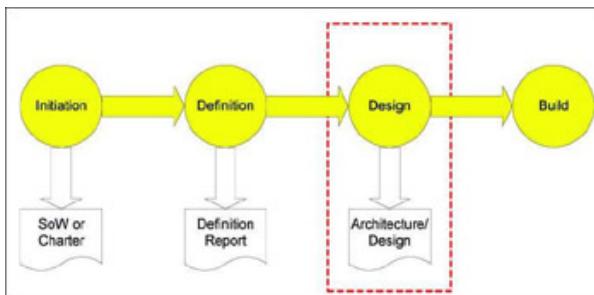


Figure 16-3: Generic implementation phases for a project

The steps are:

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.
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.
Design	This step involves designing the solution. The deliverable for this phase is the Architecture and Design document.
Build	This is where the solution is built and implemented.

16.6.1 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 (NT admin, UNIX sysadmin, 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.

16.7 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.

This book does not discuss the business process re-engineering methodologies. There are many books and Web sites that contain such information. The following redbooks may be of interest:

- *Intra-Enterprise Business Process Management*, SG24-6173
 - *Business Process Re-engineering and Beyond*, SG24-2590
-

16.8 Conclusions

This chapter has examined the issues and circumstances that affect the design of an identity management solution.

IBM Tivoli Identity Manager 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 works 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 more compact set of DSML tools that can help you get users, systems, and applications online and productive quickly.

Combining Tivoli Identity Manager with Directory Integrator provides a robust and complete identity management solution. Tivoli Identity Manager combined with Identity Director provides lifecycle management (user self-care, enrollment, and provisioning), identity control (access and privacy control, single sign-on, and auditing), identity federation (sharing user authentication and attribute information between trusted Web services applications) and identity foundation (directory and workflow) to effectively manage internal users, as well as an increasing number of customers and partners through the Internet.

The [next chapter](#) begins a more in-depth discussion of Directory Integrator.

Chapter 17: Introducing Directory Integrator

Overview

In the [previous chapter](#), we showed the need for an identity infrastructure consistent across the whole 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 through 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. Directory Integrator can be considered a *metadirectory* product, a construct that aggregates data into a single repository and synchronizes data between a set of directories and repositories. Metadirectories can also be configured to set data precedence rules and to choose different repositories as the authoritative owner for different user attributes. Gartner commented that “70% of enterprises with more than 1,000 clients will implement a metadirectory by year-end 2005.”^[1]

In this book we focus on Directory Integrator's capability to integrate and synchronize identity data across multiple repositories. Nevertheless, do not be deceived by the term metadirectory; although it contains the word *directory*, 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:

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

In the following sections we first introduce an overview and the main concepts of a directory integrator. We describe 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 5.2. However, the general concepts and many features are common to the previous releases.

[1]Source: Gartner Symposium/ITxpo 2001 May 2001 and confirmed in "Gartner RAS Core Research Note M-21-0058," J. Enck, 30 September 2003, in *Gartner Magic Quadrant for Metadirectory Products, 2H03*

17.1 Overview

In [Chapter 16, “Identity Management”](#) on [page 419](#), we discussed the business drivers for adopting a consistent identity infrastructure across a company. In 4.1, “Using a centralized user repository” on [page 84](#) 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 infrastructure have to integrate them by implementing an identity management solution. IBM Tivoli Directory Integrator is designed to fit this requirement.

Directory Integrator provides an authoritative, enterprise-spanning identity 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 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:

- No 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 clicked 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, for example, JDBC, LDAP, JMS, JNDI, and XML.
- Support of several scripting languages such as JavaScript, Visual Basic Script, and Perl Script.
- 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. It can run on UNIX (AIX, HP-UX, Solaris), Windows and Linux (RedHat, SuSE and United Linux on Intel, IBM p-series and s/390 platforms). Refer to the *IBM Tivoli Directory Integrator 5.2: Administrator Guide*, SC32-1379, for more about the supported platforms, versions, and requirements.

[Figure 17-1](#) on [page 445](#) shows a general example of an enterprise architecture using IBM Tivoli Directory Integrator. In the following section, we introduce the Directory Integrator concept and show how

information is synchronized and exchanged between the various systems.

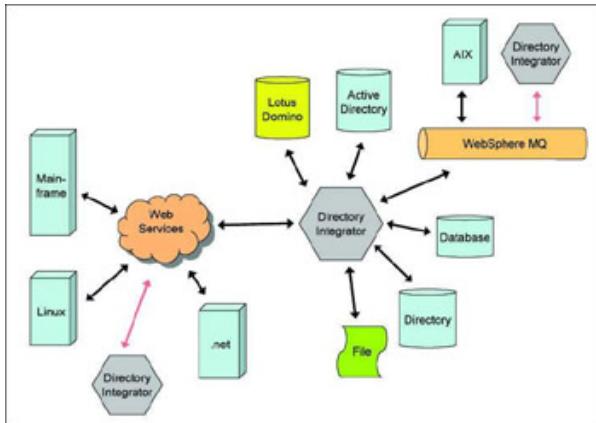


Figure 17-1: A general data integration environment

17.2 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 that trigger 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, a customer relationship management (CRM) application, the office phone system, the enterprise resource planning (ERP) system, and a messaging system with its own address book.

Data flows

These are the threads of the 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 four types of base components: *assembly lines*, *connectors*, *parsers*, and *event handlers*.

17.3 Base components

IBM Tivoli Directory Integrator is comprised of two applications:

- Toolkit IDE

This program provides a graphical interface to create, test, and debug the integration solutions. The Toolkit Integrated Development Environment (IDE) is used to create a configuration file (called a *config*), which is stored as a highly structured XML document and is executed by the run-time engine. The Toolkit IDE executable is called *ibmditk*. In 17.8, “Administration and monitoring” on [page 463](#) we describe some features of this interface.

- Run-time Server. Using a configuration file you created with the Toolkit IDE, 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 two parts:

- The *core* system, where most of the system’s functionality is provided. The core 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 Monitor 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.
- 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. There are four main types of components: *AssemblyLine*, *Connectors*, *Parsers*, and *EventHandlers*, 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 17-1](#) on [page 445](#) 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.

17.3.1 AssemblyLines

Real-world assembly lines 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 AssemblyLines receive information from various input units, perform operations on this input, and then convey the finished product through output units. AssemblyLines work on one item at a time, such as one data record, one directory entry, or one registry key.

AssemblyLines are made of a combination of Connectors, Parsers, and EventHandlers. They should contain as a 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 the AssemblyLine 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.

[Figure 17-2](#) on [page 449](#) shows an example of AssemblyLine. It is designed to update a directory with data coming from an LDIF file and a relational

database (RDBMS). A system event triggers the updates. In the following pages we develop this example to introduce all of the elements of this AssemblyLine.

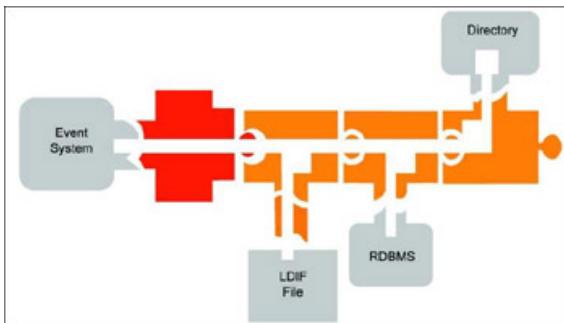


Figure 17-2: AssemblyLine

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. 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.

Another potential problem is that data sources use different types to represent stored values. IBM Tivoli Directory Integrator takes care of this as well. Everything that gets pulled into the data flow is converted to a canonical format (Java objects). As a result, attribute mapping, business rules, and transformation logic do not have to deal with type conflicts. Scripts can be added that work with this information by verifying data content, computing new attributes and values, and changing existing values until the data is ready for delivery from the line into one or more output sources. When the data is ready for output, Directory Integrator converts it back to the relevant data source-specific types.

The input and output units of an AssemblyLine are called *Connectors*, and each Connector is linked into a data store. Connectors tie the data flow to the outside world and are also where data transformation and aggregation take place. They are also where you can layer in your business, security, and identity management logic.

17.3.2 Connectors

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

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. It can be:

- An input Connector, iterating through or looking up information in its source
- An output Connector, inserting, updating, or deleting data in the connected system or device.

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.

By making your Connectors as autonomous as possible, you can also readily transfer them to your Connector Library and reuse them to create new solutions faster, even sharing them with others. Using the library feature also makes maintaining and enhancing your Connectors easier, because all you have to do is update the Connector template in your library, and all AssemblyLines derived from this template inherit these enhancements. When you are ready to put your solution to serious work, you can reconfigure your library Connectors to connect to the production data sources instead of those in your test environment, and move your solution from lab to live deployment in minutes.

Whenever you need to include new data to the flow, simply add the relevant Connector to the AssemblyLine. In the example of [Figure 17-2](#) on [page 449](#) we see three connectors: two input connectors to an LDIF file and to an RDBMS, and one output connector to a directory.

IBM Tivoli Directory Integrator provides a library of Connectors to choose from, such as Lightweight Directory Access Protocol (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 one of the leading scripting languages, including JavaScript, VBScript, and PerlScript. You can also create your

own, either with a scripting language inside the Script Connector wrapper, or originate with Java or C/C++.

Furthermore, IBM Tivoli Directory Integrator supports most transport protocols and mechanisms, such as TCP/IP, FTP, HTTP, and Java Message Service (JMS)/message queueing (MQ). It also supports secure connections and encryption mechanisms as shown in 17.4, “Security capability” on [page 455](#).

[Table 17-1](#) 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 5.2: Reference Guide*, SC32-1377.

Table 17-1: Main available connectors

Applications	PeopleSoft, IBM Tivoli Access Manager (through database access, scripting, or API calls)
Databases (using ODBC, JDBC)	Oracle, Microsoft Access and SQL Server, IBM DB2 and Informix®
Directories (using LDAP)	CA eTrust, Critical Path, IBM, iPlanet, Microsoft Active Directory and Exchange, Nexor, Novell, OpenLDAP, Oracle, Siemens
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
Changes & Deltas	LDAP Changelog, Active Directory changes, NT/AD Password sync, TCP connections, HTTP gets and posts

17.3.3 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 17-3 on page 452](#), a Parser is used to interpret and translate information from an LDIF file. The extracted information is converted to a Java object with a canonical data format so that the LDIF Connector can work with this object and dispatch it along the AssemblyLine.

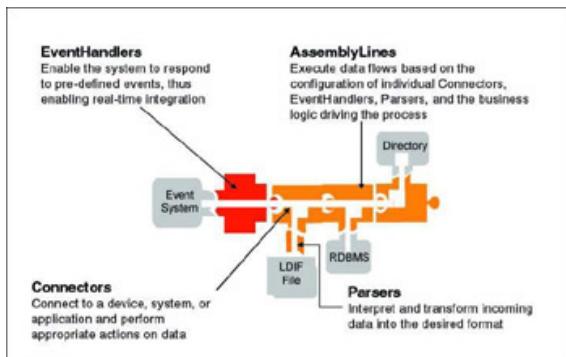


Figure 17-3: AssemblyLine with Connectors, Parsers, and EventHandlers

17.3.4 EventHandlers

EventHandlers provide functionality for building real-time integration solutions.

As with Connectors, EventHandlers can have data source “intelligence” that enables them to connect to a system or service and wait for an event notification. Examples are the Mailbox EventHandler, which can detect when new messages arrive in a POP3 or IMAP mailbox, and the LDAP EventHandler, which can catch changes made to a directory. When an event occurs, the EventHandler stores the specifics of the event and then performs logic and starts AssemblyLines according to the condition or action rules that you set up.

Sometimes Connectors can also be used to capture events, as is the case with the JMS (MQ) Connector or the LDAP Changelog Connector, both of which can be configured to wait until new data appears and then retrieve

it. However, because the EventHandlers operate in their own thread, they can be used to dispatch events to multiple AssemblyLines. This provides a cleaner and more straightforward method of filtering and handling multiple types of events from the same source (such as SOAP or Web services calls). EventHandlers can also be configured for auto start, meaning that if you start up a server, these EventHandlers will be activated immediately.

Figure 17-3 shows that a system event can trigger the AssemblyLine.

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

17.3.5 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. IBM Tivoli Directory Integrator automatically calls these user-defined functions as the AssemblyLine runs.

The majority of the scripting in IBM Tivoli 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.

17.3.6 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 be anywhere in the system to add or modify the components of an AssemblyLine. Connectors, Parsers, EventHandlers, 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 a number of plug-in scripting languages and extensive script libraries. The most commonly used are JavaScript, VBScript, and PerlScript.

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.

17.3.7 Password synchronization

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 17-4](#) 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 IDS Changelog and starts an assembly line when a change notification is received.

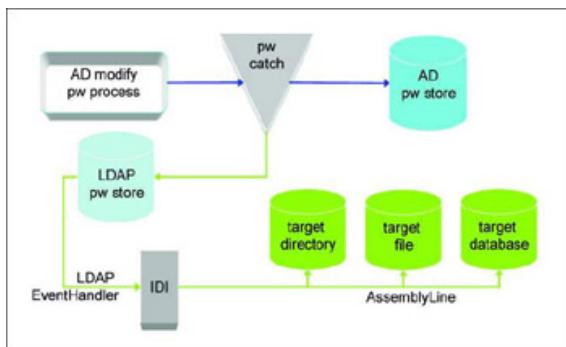


Figure 17-4: Password interception with Active Directory

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.

17.4 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 queueing (MQ). SSL and other encryption mechanisms can be added to any of these methods to secure the information flow. Additionally, the graphical interfaces (IDE and AMC) can be configured to be accessed by SSL. SSLv3 encrypts communications on the wire. 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.

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.

17.5 Physical architecture

We cannot show all possible architectures with all of the different data sources and data flow, so we introduce some general considerations about the use of an enterprise directory and some basic structures of data flow. We assume that readers understand the capability of IBM Tivoli Directory Server and be able to design their own environment.

17.5.1 Combination with an enterprise directory

There are two approaches to integrating existing enterprise data stores and building an authoritative source for identity information:

- Introduce one main directory and then synchronize and publish data from there to all other repositories.
- Avoid the central repository and configure automatic data flow and reconciliation between the repositories.

IBM Tivoli Directory Integrator supports both solutions and can provide valuable assistance. By design it seems especially suited for the second approach. A scenario based on this architecture is shown in [Figure 17-1](#) on [page 445](#). 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 17-5](#), 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 17-5](#) 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.

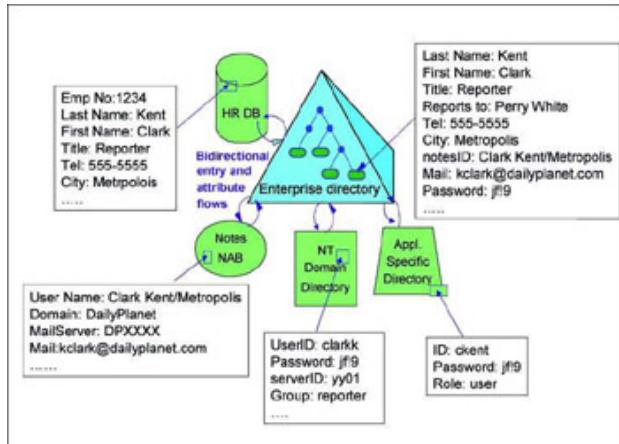


Figure 17-5: Scenario with an enterprise directory

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, 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. In the [next section](#) we discuss different topologies available for data flows.

17.5.2 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. According to 17.3.4, “EventHandlers” on [page 452](#), every data flow can be triggered by specific events or configured on a time schedule. It is not necessary to investigate on how our example flows are triggered.

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 an 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 17-6](#). 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.

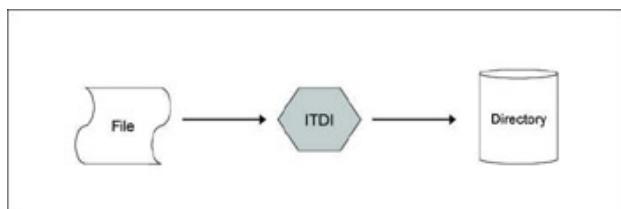


Figure 17-6: One-to-one integration

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. Directory Server 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.

Many-to-one

The second scenario is shown in [Figure 17-7](#) on [page 458](#). 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.

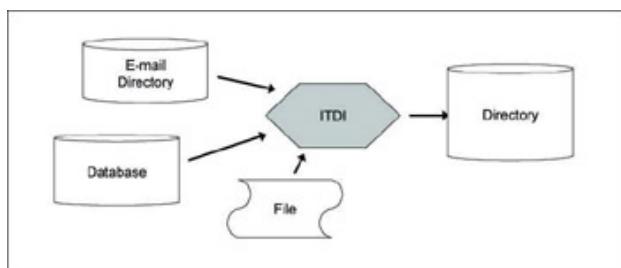


Figure 17-7: Many-to-one integration

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.

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. 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 17-8](#) on [page 459](#), 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 Directory Server 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 17-8](#), 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.

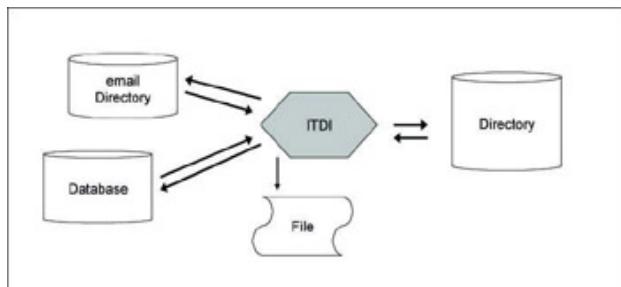


Figure 17-8: One-to-many integration

17.5.3 Multiple servers

In the scenarios shown so far, there is only one IBM Tivoli Directory Integrator server. There are several reasons why a company would prefer to have multiple server instances. These servers can communicate among each other using various mechanisms such as message queuing, HTTP, e-mail, FTP, and Web services. Some reasons why an architecture with multiple servers might be desirable are:

- Distance. 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.
- Security. For instance, data might have to pass through firewalls that block protocols such as LDAP and database access. In this case a Directory Integrator server might be placed on each side of the firewall and the two servers could use a channel allowed by the firewall such as MQ or HTTP protocol. Another example could be that different business units want to retain local control over information shared with others. Local configuration enables administrators to set restrictions on the data sets that are exposed, the attributes that are sent and received, and any local transformation rules that have to be applied to the data going to or coming from the other participants.

- Availability and scalability. We address this topic in 17.6, “Availability and scalability” on [page 460](#).

Figure 17-9 depicts a generic example of an architecture with multiple servers. 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.

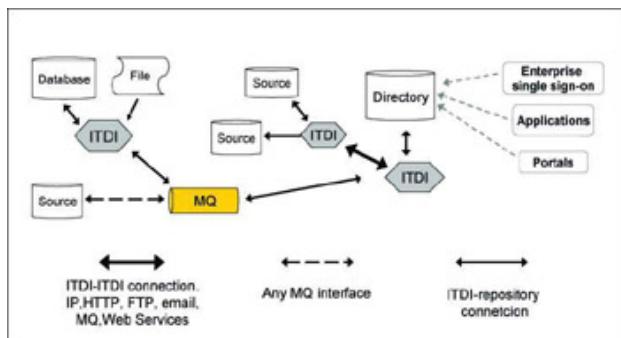


Figure 17-9: Multiple server environment

In the following section we introduce another level of complexity by using multiple servers to implement high availability and increase performance.

17.6 Availability and scalability

The basic concept of high availability is to have at least two servers capable of performing the same job and a fail-over mechanism to switch from one server to the other if one of the servers fails.

IBM Tivoli Directory Integrator does not provide an internal fail-over mechanism. Therefore, one way to provide automatic high availability is to install the server in a cluster environment such as HACMP for AIX. Otherwise, manual methods can be implemented to replace a failing server. However, remember that all AssemblyLine definitions and configurations are stored within one XML file called Config. Therefore, if a server fails, it is sufficient to start a separate server with the same Config.xml file in order to continue the service. 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. [Figure 17-10](#) shows a high availability solution. A fail-over mechanism must be configured between the two servers, depending on functional requirements of the data integration environment.

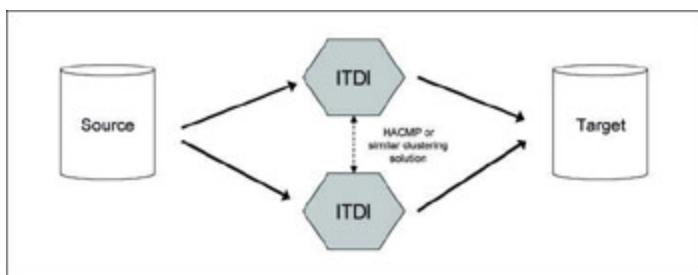


Figure 17-10: High available servers

Scalability is a strong feature of Directory Integrator. There is virtually no limit to the number of servers that can be

added. Different servers can work on different data flows, as shown in [Figure 17-9](#) on [page 460](#), or on different data of the same data flow, as shown in [Figure 17-11](#) on [page 462](#). 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.

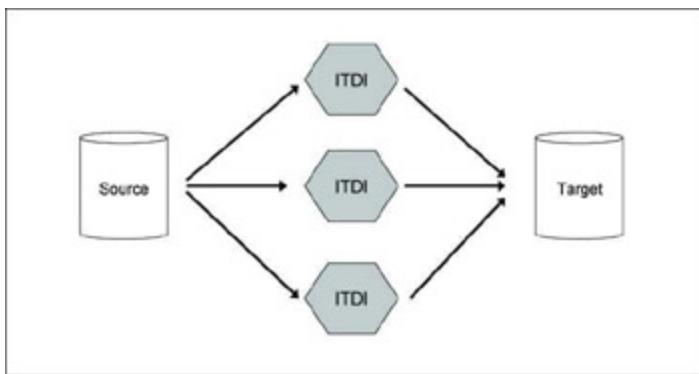


Figure 17-11: Scalable servers ITDI

17.7 Logging

IBM Tivoli Directory Integrator enables you to customize and size logs and outputs. It relies on log4j as a logging engine. It is a very flexible framework that enables logging to individual files, the eventlog, and syslog, and it can be tuned so that it suits most needs. In addition to built-in logging, script code can be added in AssemblyLine to log almost any kind of information.

Key data is logged from the Directory Integrator engine, from its components (Connectors, Parsers, and so on), as well as from user's scripts. Configuring the logging can be done globally or specifically for each AssemblyLine, EventHandler, and Server. Almost every Connector has a parameter called Debug Mode, 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. Different information can be dumped, such as the content of an 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 5.2: Reference Guide*, SC32-1377, for more details and examples.

In addition, the AssemblyLine debugger enables suspension of AssemblyLine processing and shows values of various variables and objects in a debugger console. By adding watch variables to the Watch List, the administration tool displays the value of the variables at all enabled break points. You can watch variables available within the script engine or run small scripts.

17.8 Administration and monitoring

In 17.3, “Base components” on [page 447](#), we introduced the Toolkit Integrated Development Environment (IDE). This graphical tool enables you to create, test, and debug any AssemblyLines with all the components and the optional scripting. 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. When the AssemblyLines are ready and the integration solution is deployed, administration can be performed either from a command line or with a graphical interface.

The command line utility is called *Admin Tool* and is based on the **ibmditk** command. This accepts many flags and parameters that enable the necessary tasks, such as selecting a configuration file and managing individual AssemblyLines and EventHandlers.

The graphical interface is a Web console called the Administration and Monitor Console (AMC). It enables management and monitoring of the active AssemblyLines through a Web browser. AMC users can have different privileges:

- The administrator role can use each and every feature of AMC.
- The operator role can view all AMC output, details for all components, and start or stop AssemblyLines or EventHandlers. However, it cannot perform administrator operations, such as:
 - Change or save an IBM Tivoli Directory Integrator configuration.

- Import components.
- Reload or shut down IBM Tivoli Directory Integrator Server.
- The user role can access all AMC screens from the main menu, but they cannot perform any actions:
 - Cannot start and stop AssemblyLines or EventHandlers.
 - Cannot update the IBM Tivoli Directory Integrator configuration.
 - Cannot view logs and details for components such as parameters and statistics.

Finally, to prevent some headaches, we add these two notes:

- AMC does not provide configuration isolation between users, as IBM Tivoli Directory Integrator Server operates with a single configuration. All Administration and Monitor Console users connected to a Server at a given time work on and see the same configuration file.
- It is not possible to work on the same Server (and operate on the memory representation of the Server's configuration) from Admin tool and AMC.

Whenever an AssemblyLine or EventHandler is activated from the Admin tool, a new system process is started that runs the IBM Tivoli Directory Integrator Server with the AssemblyLine or EventHandler that have been chosen; AMC is disabled for the correspondingly started Servers.

17.9 Conclusions

In this chapter we introduced 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.

In the [next chapter](#) we introduce IBM Tivoli Identity Manager. This product, which focuses more on the user and identity rather than on general data, includes IBM Tivoli Directory Integrator and can be used for identity feed and data synchronization. It provides a more complete set of functionality to manage identity credentials, workflow, and audit.

[Chapter 20, “Synchronizing the enterprise” on page 501](#), 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.

Chapter 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

18.1 Lifecycle management

As discussed in previous chapters, identity management is the process of managing users and their accounts across all systems. User lifecycle management is the process through which identities are created, managed, and ultimately destroyed. [Figure 18-1](#) graphically represents the lifecycle management phases.

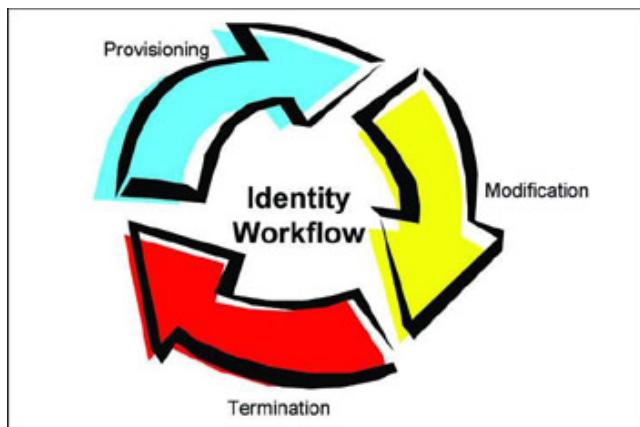


Figure 18-1: User lifecycle management phases

The provisioning cycle begins the process. Here the system provides automated process and IT policy enforcement using a coordinated approach to access resources and the associated privileges.

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.

18.1.1 The provisioning cycle

The provisioning cycle includes:

- 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.

18.1.2 The modification cycle

During the maintenance phase of the life cycle, administrators maintain the following elements:

Identity	The user's credentials, such as user name and password, as well as information about the user, 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 life cycle offers significant opportunity for automation and efficiency gains.

18.1.3 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 not uncommon.

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.

18.2 Logical component architecture

The logical component design of Identity Manager may be separated into three layers of responsibility, which are depicted in the center of [Figure 18-2](#).

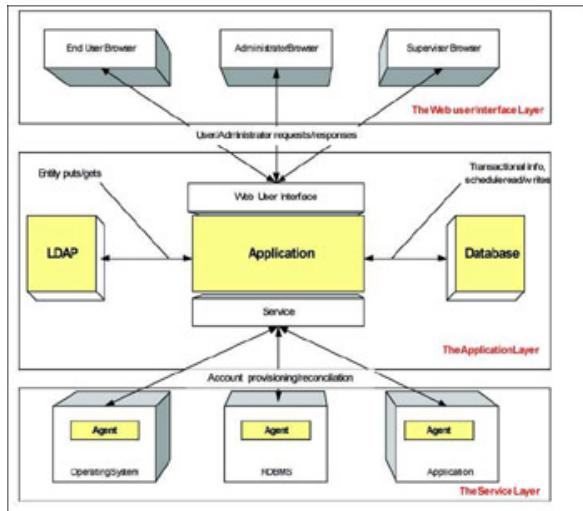


Figure 18-2: Identity Manager logical architecture

They are:

- The Web User Interface layer
- The Application layer
- The Service layer

This diagram depicts the graphical workings of the package.

18.2.1 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](#) on [page 468](#), there are three types of user interaction points: end 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. A brief description of each module follows [Figure 18-3](#).

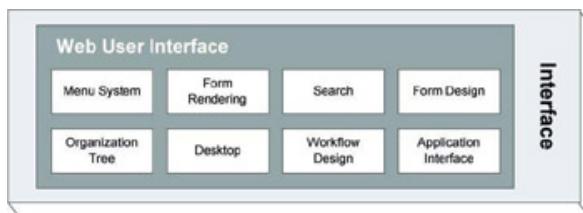


Figure 18-3: Web User Interface module subprocesses

Menu system module

The Menu System module provides consistent menu and short-cut mechanisms that are used for navigation throughout the user interface.

Organization tree module

The Organization Tree module provides the graphical tree representation of the organizational structure in which entities managed through the user interface are logically stored.

Form rendering module

The Form Rendering module provides the run-time interpreter to display the customized forms designed in the Form Design module.

Desktop module

The Desktop module provides a framework for providing a consistent layout in the pages displayed in the user interface. It is within this framework that products of the other Web User Interface modules are displayed, such as the Menu System and Organization Tree.

Search module

The Search module provides a framework for general and more specific search interfaces to be used throughout the user interface.

Workflow design module

The Workflow Design module provides a graphical workflow design environment. A workflow process consists of a set of activities that are executed in an ordered fashion according to conditional transitions. The designer provides a graphical way of defining such a workflow process. This module makes use of applets for its more flexible demands.

Form design module

The Form Design module provides a near WYSIWYG (What You See Is What You Get) user interface design environment for customizing the forms that display information about the entities managed through the user interface. This module makes use of applets for its more flexible demands.

18.2.2 Application layer

The core of the 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. The Applications module provides the external interface to the provisioning platform.

We can break these areas down into additional subprocesses, as [Figure 18-4](#) shows.

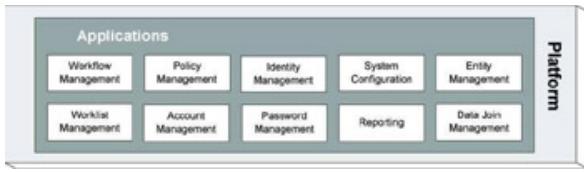


Figure 18-4: Applications Module subprocesses

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.

Worklist management module

The Worklist Management module provides the capabilities required to manage an individual's workflow-driven worklist or assignments. This includes the approval or disapproval of a request or the fulfillment of a Request For Information (RFI).

Policy management module

The Policy Management module provides the capabilities to manage the policies in the system, including those for provisioning, passwords, service selection, and identity.

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.

Password management module

The Password Management module provides the capabilities required to change or synchronize passwords according to a defined set of password

policies or rules.

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 life cycle of the entity type.

Data join management module

The Data Join Management module provides the capabilities required to manage the data join features of the system. This includes the creation, modification, and deletion of data flows to be followed for joining data between changing identities and accounts.

18.2.3 Service layer

If the 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 communicates with the user-facing Web server, the agents residing on the managed services, and 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 make use of other services to achieve their goals. A brief description of each module is given next, as well as the graphical overview shown in [Figure 18-5 on page 473](#).

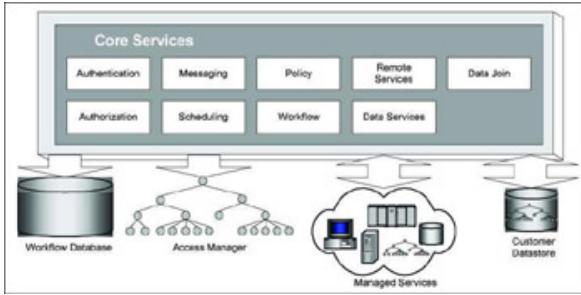


Figure 18-5: Core Services module subprocesses

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.

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 and 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 enables 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. Metadata information about the persistent data can also be retrieved using this module.

Data join module

The Data Join module provides the join engine capabilities of the platform, including the distribution of changes between related identity

and account entities in the system. This module works at a lower level than the provisioning logic, so it is not audited the way other provisioning transactions are. The join operations are triggered as a result of the completed provisioning transactions.

18.2.4 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.

18.2.5 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.

18.2.6 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 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 agent is provided. The agents typically use an XML-based protocol, Directory Services Markup Language (DSML), as a communications mechanism.

18.2.7 Directory Services Markup Language connectivity

DSML provides a method for processing structured directory-based information as an XML document. DSML is a simple XML schema definition that enables directories to publish profile information in the form of an XML document so that it can be shared easily via IP protocols such as HTTP/S, as shown in [Figure 18-6](#) on [page 476](#).

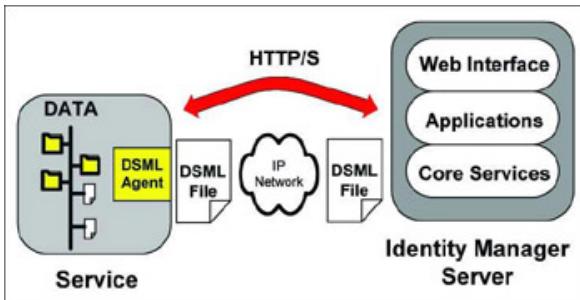


Figure 18-6: DSML connectivity to a service

Transactions from the Identity Manager server are sent securely via HTTPS to the service agent and then processed by the agent. An 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 agent. The agent deposits the new information within the application or operating system that is managed.

IBM Directory Integrator connectivity

IBM Directory Integrator is a data synchronization tool that can be used for manipulating data from a variety of identity sources and storing and updating that identity information in Identity Manager. Directory Integrator synchronizes data residing in directories, databases, e-mail systems, operating systems, and application systems such as HR, ERP, and CRM.

Directory Integrator is a metadirectory tool that facilitates synchronizing and exchanging information between sources. The plug-and-play functionality of these components facilitates rapid prototyping and implementation of intelligent data flows.

Two different models exist for metadirectories:

- The Meta-View, where all data is aggregated at a central store and operations against it are pushed back to the authoritative source
- Point-to-Point Synchronization, where events drive data updates based on business rules and technical requirements

Which model is appropriate is based on business needs and requirements. Directory Integrator supports both models and design will drive the final choice.

Directory Integrator provides a rich scripting environment to adapt to your business rules, and it utilizes DSMLv2, which is an industry standard for directory manipulation. IBM Directory Integrator is now part of the Tivoli Integrated Identity Management family.

Using IBM Tivoli Identity Manager's JNDI API, it is possible to connect to just about any service, application, or endpoint, as shown in [Figure 18-7](#).

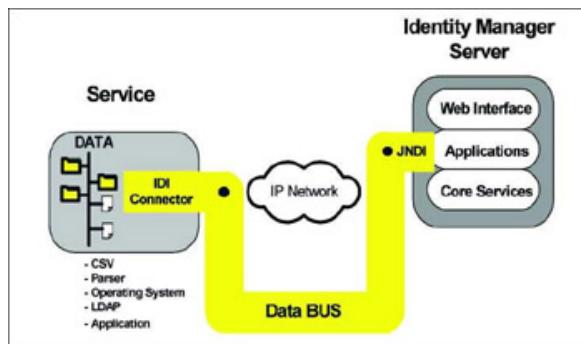


Figure 18-7: IBM Directory Integrator service communication

By using Directory Integrator and IBM Tivoli Identity Manager, communications to resources such as a CSV file or a Microsoft Excel spreadsheet containing contractor employment details may be possible through a parser connector. Identity Manager would see this as a service that it manages and that can be incorporated into its workflow.

18.3 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.

Chapter 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 process 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 zones

As we have discussed throughout this book, network zones play an important role in an Identity Manager design:

- Uncontrolled (the Internet)
- Controlled (Internet DMZ)
- Controlled (intranet zone)
- Restricted (production zone)
- Secured (management zone)

Internet (uncontrolled)

Nothing is placed in this zone.

Internet DMZ (controlled zone)

The Internet DMZ is a controlled zone that contains components that clients may directly communicate with. 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

No Identity Manager components should be placed in this zone.

Intranet (controlled zone)

As stated above, a controlled zone contains components that clients may directly communicate with. In this case, it is the corporate intranet that acts as a buffer to the production zone. In this area, you may find internal-use Web servers that may be accessed freely. While it is not advisable, it is possible to place certain Identity Manager components here.

Production (restricted zone)

One or more network zones may be designated as restricted, meaning that they support functions where access must be more controlled. A point to remember is that direct access from an uncontrolled network should not be permitted. A restricted network is typically bounded by one or more firewalls, and incoming and outgoing traffic may be filtered as appropriate. IBM Tivoli Identity Manager components found in this location could be Web server and

directory (LDAP) replicas. While it is possible to locate additional systems here, it is not strictly advisable.

Secured (management zone)

This area is strictly controlled, with access available to a small group of authorized staff. As with production zones, it is possible to have one or more secured zones. However, access to one security area does not necessarily give access to another secured area. Identity Manager components found in this location would include Identity Manager server, Identity Manager database, and the directory (LDAP) master.

19.1.2 Other 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 482](#) summarizes the general Identity Manager component type relationships to the network zones discussed above.

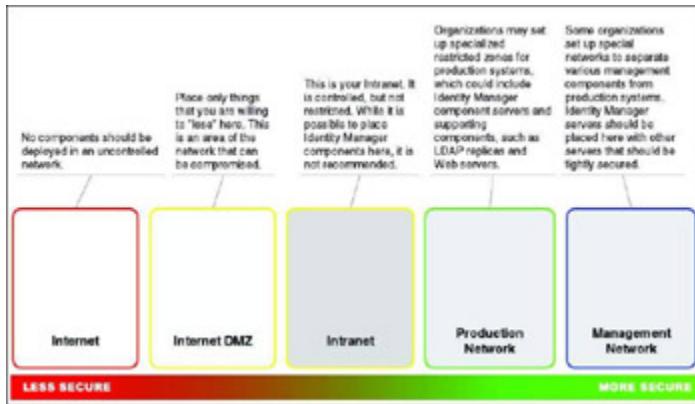


Figure 19-1: Network zones for Identity Manager placement

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. An exception to this may be to place one or more GUI servers 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 MASS. There are many models that may be constructed. This is meant to be a *best practice*. Walkthroughs of complete business processes, including exceptions, and help you to create a viable solution and refine the requirements.

19.2 An Identity Manager scenario

In earlier chapters, we discussed Stocks-4U.com. For this example, we are going to talk about another company - Areally Big Investment Security Corporation.

Company profile

Areally Big Investment Security Corporation is 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. Historically, Areally Big Investments has operated with little Internet presence, preferring instead to offer personalized broker services to their customers as well as handling transactions for the various stock markets around the world. Because there are numerous operating regulations that have been imposed by various government authorities, Areally Big Investment has been reluctant to offer more than informational services on their Web sites. With offices located around the globe, Areally Big Investments has 70,000 employees who have access to financial data systems, e-mail, and intranet applications, as well as an additional 10,000 outside banking partner users who have access to certain systems via VPN only.

There are several mini data centers 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 Investments currently has a robust security and network infrastructure in place. These systems are audited regularly, and risk assessments are scheduled quarterly.

However, Areally Big Investments has been aware for some time that managing their information infrastructure more efficiently could realize a significant cost savings. After examining operational costs and conducting a cost analysis study, the bank has 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 package that will best fit their needs.

19.2.1 Business requirements

To reduce overall IT operational costs and to centralize user management, Areally Big Investments has 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.

The corporate vision is to continue to increase employee productivity and prevent customers from becoming dissatisfied, while reducing overall cost of operations. During the cost analysis study, the need to apply uniform security policies 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 security management
- Reduced costs
- Increased efficiency
- Single sign-on and unified user experience
- Consolidated user management
- Single authoritative data source
- Ease of compliance with regulations and audit requirements

19.2.2 Designing the solution

Given the mandate of security, efficiency, and productivity, and our more detailed list above, 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? 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, managed systems and applications, policies, and workflow.

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](#) on [page 485](#) gives an idea of how the process should begin to come together.

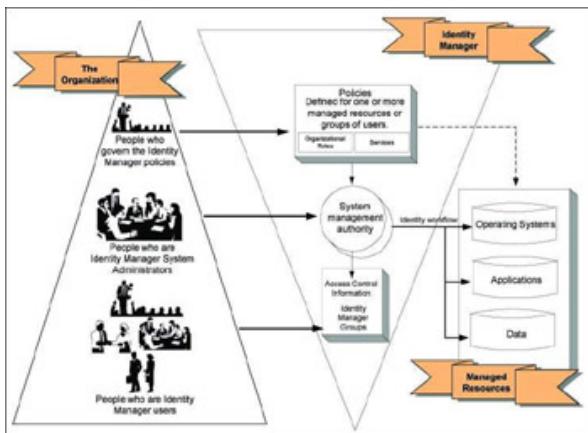


Figure 19-2: IBM Tivoli Identity Manager relationships

After gathering all of this information, you can begin to design the required workflows. [Figure 19-3](#) on [page 486](#) shows a generalized workflow diagram. Diagrams are useful for mapping the process of creating or modifying identities.

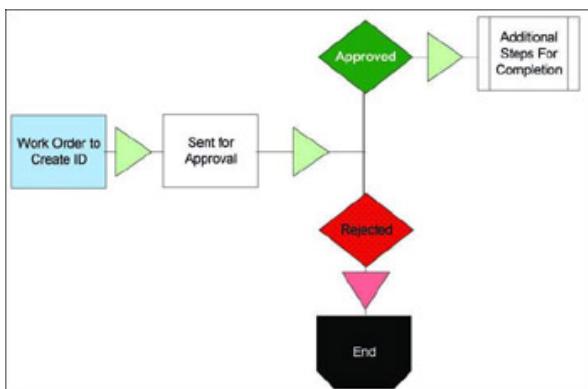


Figure 19-3: A workflow diagram example

How to collect the criteria and in what order to collect them will be defined by the project or by utilizing a specific methodology, such as the IBM Global Services Methodology. For this scenario, we use very basic requirements for identity management.

A really Big Investment Security Corporation has set the criteria for the implementation of IBM Tivoli Identity Manager. First on the list of tasks to accomplish is a single password for user accounts. 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 wish 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 19-4 on page 487](#) shows a sample basic Identity Manager architecture, given the infrastructure we currently have and the MASS parameters we need to apply.

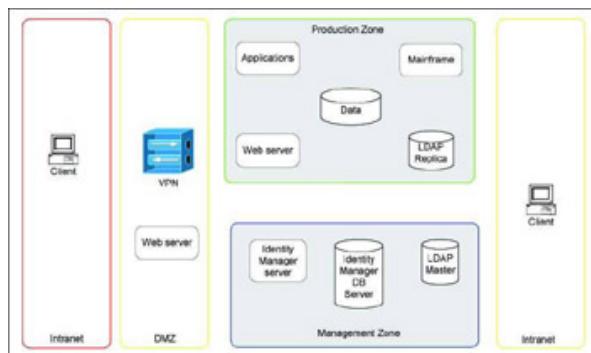


Figure 19-4: Basic physical architecture for A really Big Investment Security Corporation

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 agents and Web server reside in the production zone. [Figure 19-5 on page 488](#) shows the data map for the architecture. Note that after communicating the necessary identity information to the agents, access for users is granted or denied to the resource based on the policy information. If access is granted,

interaction is conducted directly with the resource, not the Identity Manager agent.

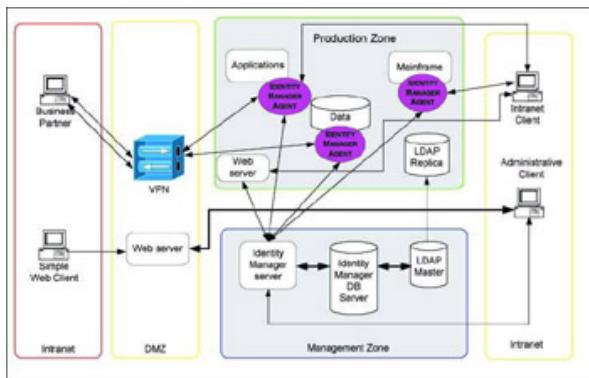


Figure 19-5: A really Big Investment Security Corporation basic architecture

19.3 Integrating with Access Manager

As we discussed earlier in this book, 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 19-6 on page 489](#).



Figure 19-6: 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. Some of these tasks have been automated and are provided in a collection of utilities called the Provisioning Fast Start collection, which is included on the *IBM Tivoli Access Manager Base CD*.

The integration of Identity Manager with Access Manager requires that we consider the placement of all of their combined components. [Figure 19-7 on page 490](#) shows, by zone, the recommended placement of Access Manager components.



Figure 19-7: Network zones for Access Manager placement

[Figure 19-8 on page 491](#) shows a sample architecture for integrating Identity Manager and Access Manager. The data flows are not represented in this example as its intent is to demonstrate server placements.

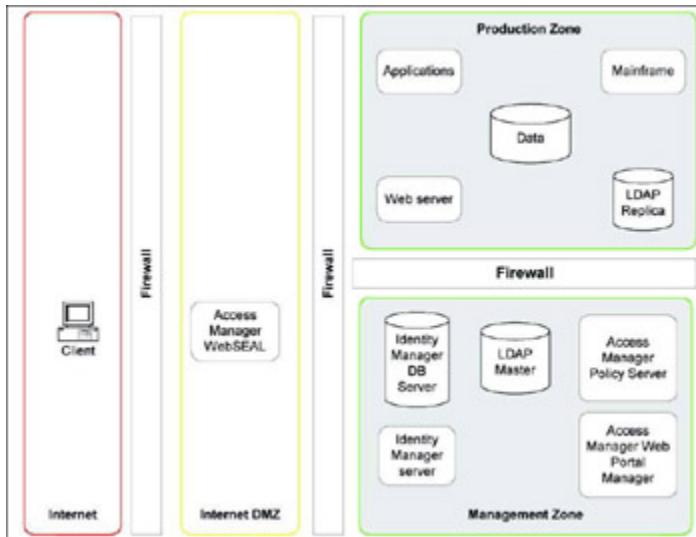


Figure 19-8: Integrated architecture for Access Manager and Identity Manager

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.

To integrate the environments, you must add a Tivoli Access Manager service to Identity Manager, so that Identity Manager can manage the Access Manager accounts. Each system that is managed by Identity Manager must be assigned to Identity Manager as a *service*. If Identity Manager has to manage more than one Tivoli Access Manager system, you should create a service for each Tivoli Access Manager system. If your Access Manager environment includes resources that permit global sign-on access (that is, GSO resources and GSO resource groups), be sure to install the Tivoli Access Manager GSO Agent. This agent enables you to create services for GSO resources and GSO resource groups.

19.3.1 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
- Importing user data into Tivoli Identity Manager from an existing Tivoli Access Manager environment or an existing corporate directory
- 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

19.3.2 Importing and synchronizing user data

Identity Manager is designed to be a central location for corporate identity management. However, in your environment, other IBM Tivoli security applications with user management (such as Access Manager) might have been installed already and will have to co-exist with Identity Manager. Consequently, several user data records might exist for the same user.

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.

If Access Manager or other applications with user data records co-exist with Identity Manager and up-to-date user attributes are needed for these applications, Tivoli Identity Manager data must be dynamically synchronized with the user records in these applications.

IBM Tivoli Directory Integrator AssemblyLine utility

The IBM Tivoli Directory Integrator AssemblyLine samples utility is included in the Provisioning Fast Start collection on the *IBM Tivoli Access Manager Base CD*. The utility uses IBM Directory Integrator to import Tivoli Access Manager and corporate directory users to Identity Manager and to synchronize Tivoli Identity Manager user attributes with those in Tivoli Access Manager.

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.

Keep in mind, this book focuses on architecture and not on the specifics of installing or configuring the environments for use. If you require more explicit technical information about integration, see:

- *IBM Tivoli Access Manager for e-business IBM Tivoli Identity Manager Provisioning Fast Start Guide Version 5.1*, SC32-1364
- *IBM Tivoli Identity Manager Access Manager Agent for Windows Installation Guide Version 4.5*, SC32-1165

19.3.3 Integrated architecture with IBM Tivoli Identity Manager agents

When the services are connected and integrated, Identity Manager becomes the focal point for all user requirements, and Access Manager is the focal point for access requirements. Note that in [Figure 19-9](#) on [page 494](#), Identity Manager Server talks to the agents in the Access Manager server environment and not to the Access Manager servers themselves. Access Manager is a service of IBM Tivoli Identity Manager. Identity Manager can then utilize the information that is stored in the Access Manager server environment. (For simplification purposes, this example does not show the interconnection to managed resources with Identity Manager.)

In other words, Identity Manager has the capability to make password changes to the Access Manager environment. WebSEAL also has the ability to check the strength rules set in Identity Manager and initiate the change and propagate it to other accounts managed by Identity Manager. This ensures that regardless of whether users change their password through Identity Manager or WebSEAL, password synchronization can be maintained.

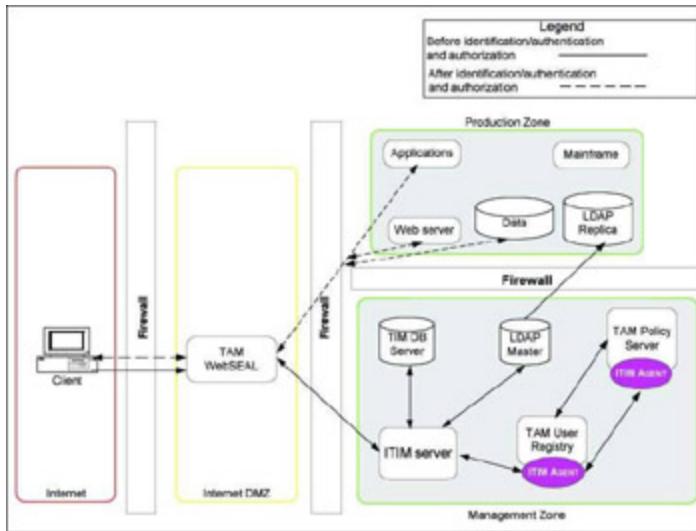


Figure 19-9: Sample Identity Manager and Access Manager integrated architecture

19.4 Access Manager and Directory Integrator scenario

We now return to our fictional company Stocks-4U.com. As we know, they already have a robust Access Manager implementation in place and they have utilized many of the Access Manager custom features such as the aznAPI and tag values to create their own user management screens and login and logout features. They have also implemented advanced features such as cross-domain high availability, CDSSO, and so on, enabling them to manage their infrastructure and Web users securely.

As of today, Areally Big Investment Securities Corporation has no Web accessible application systems in place but only a Web site with information and static content. As we know, they do have Identity Manager deployed internally to manage their employee user accounts and their VPN partner users.

For some time, Areally Big Investments has been exploring expanding into the Web investments market. Historically, their focus has been on “brick and mortar” operations offering brokerage full service and personalized attention to customer accounts.

Being a security, as well as a cost-conscious company, Areally Big Investments made the decision to acquire, rather than create from scratch, a Web investments business.

After lengthy negotiations, Stocks-4U.com and Areally Big Investments have agreed to a merger. Both companies realize that they complement each other in a way that should be profitable and promote increased revenue growth well into the future. This merger has also included the decision to maintain the Stocks-4U.com name and Web presence while attempting to integrate with Areally Big Investments’ well-ordered identity workflow and business processes. The primary objective is to maintain, and in the future contain, the substantial

infrastructure investments that both have made. The secondary objective is to preserve the unique qualities and strengths of each company in the investments market.

19.4.1 Business requirements

To reduce overall IT transition costs, the current infrastructure of each company must be utilized. Both companies have established that mitigating security risks is essential to the final outcome. The combined corporate vision is to continue to increase Web availability as well as employee productivity and prevent customers from becoming dissatisfied while reducing overall cost of operations. However, maintaining an authoritative data source at Areally Big Investments is crucial. An added requirement is that certain individuals within Areally Big Investments must be able to access Stocks-4U.com and vice versa.

After lengthy discussions, access and authorization decisions (Access Manager) are to be kept separate from identity management, thus keeping Web security completely outside of help desk and identity areas (Identity Manager).

Point-to-point synchronization should occur where events drive data updates based on business rules and technical requirements.

19.4.2 A solution

Maintaining the two separate environments can be achieved using IBM Tivoli Directory Integrator to merge the data from two (or more) sources. By utilizing Directory Integrator, applications are not affected, synchronization is done at the data layer, and latency is minimal. (Directory Integrator can be scheduled to run as needed.) While the mandate seems unwieldy, reducing the problem into smaller manageable pieces is the best approach.

We now look at the basic architectures of each company. As [Figure 19-10](#) shows, because they were both created using MASS, they are identical in their conceptual architectures. Each has an Internet DMZ, a production zone, an intranet, and a management zone in their environments.

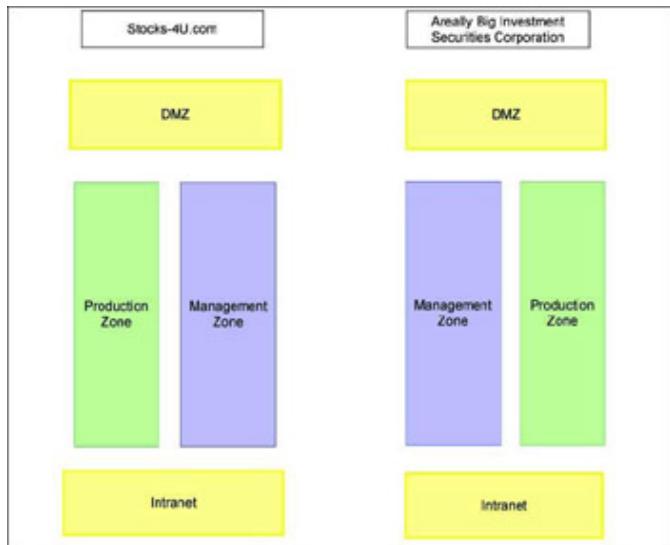


Figure 19-10: Basic high-level architecture compared side by side

Next, we look at the basic component placements. [Figure 19-11](#) shows that the only items outside of the management zones are WebSEAL at Stocks-4U.com and the IBM Tivoli Identity Manager Web server at Areally Big Investments.

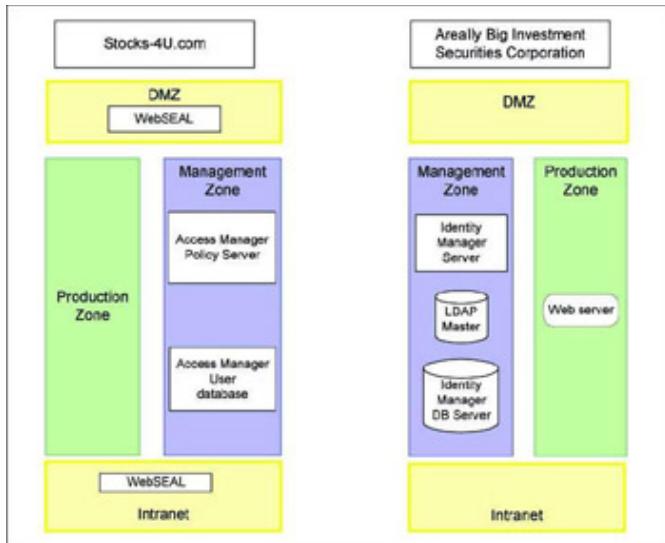


Figure 19-11: Server component placement in each environment

If we are to maintain an authoritative data source at Areally Big Investments and allow access to people in both organizations, we need to supply a bidirectional data feed from each location that is synchronized. [Figure 19-12 on page 498](#) shows the relationship that must be established between the two environments.

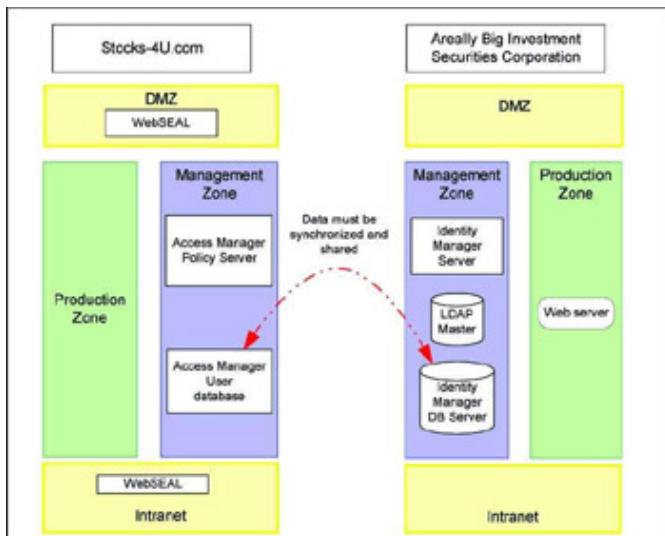


Figure 19-12: Data synchronization focus

Now that we know what we have to link, we need to place Directory Integrator. Because Directory Integrator is a component of IBM Tivoli Identity Manager, and we understand the requirements for the location of our authoritative data source, it makes sense to place it within the management zone of Areally Big Investments. (It could also reside inside the management zone of Stocks-4U.com.)

Integration is about communication. Specifically, what systems and devices must participate? When do they need to communicate? What are they communicating to each other and how are they communicating it? So far, we have identified that Stocks-4U.com and Areally Big Investments must communicate their IBM Tivoli Identity Manager and Access Manager user information with each other. When they need to communicate is predicated on the business drivers—will one daily update fulfill the company needs or does the synchronization need to happen more frequently? How they will communicate is still unanswered.

Although Directory Integrator makes building data flows fast and easy, the quality of the resulting solution is dependent on how good your specifications are.

Directory Integrator helps the process by removing the platform and vendor technology blinders that block our vision and limit the imagination. When you approach an integration problem at the data-flow level, you reduce complexity. As you start to think in terms of simplify-and-solve, you see possibilities for additional integration. You also begin to view the complexity of data integration from a completely different perspective.

[Figure 19-13](#) shows how reducing the complexity and architecture enables you to see how the data flows will be realized.

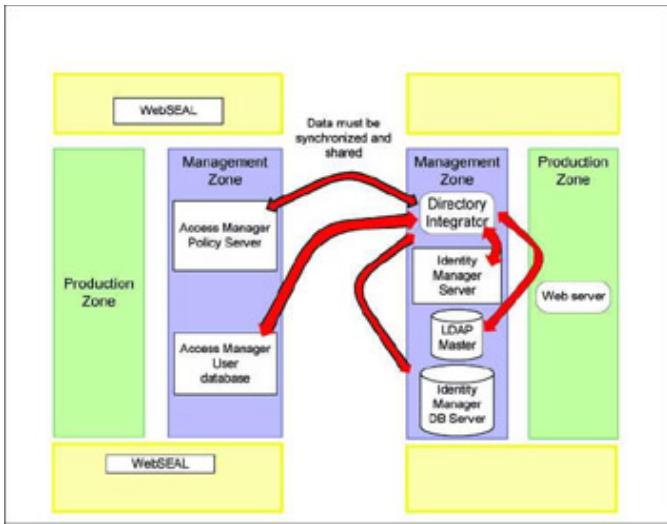


Figure 19-13: Directory Integrator data flows

How does the data actually pass through the two networks using Directory Integrator? There are several ways to achieve this. With the current network configurations, we could install an additional WebSEAL in the DMZ of Areally Big Investments. Alternatively, we could utilize the VPN that is currently available at Areally Big Investments and the WebSEAL at Stocks-4U.com. Or, we could utilize an all new VPN communicating via SSL between the two management zones of both companies.

Whatever the approach for the connectivity, this example's focus is on how to integrate two very different environments with complex business drivers, requirements, and constraints. By breaking the problems down into micro, and not attempting to encompass the macro environment's every nuance, the problem of integrating becomes an opportunity for success.

19.5 Conclusion

Alone, IBM Tivoli Identity Manager is a powerful collection of tools that for managing the lifecycle of a user ID from many different facets. When coupled with IBM Tivoli Access Manager, a federated identity management solution becomes available for the enterprise.

Identity Manager and Access Manager offer extensive opportunities to solving complex security needs and requirements. Both packages can stand alone or be combined for greater control of your environment.

Chapter 20: 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 MASS (Method for Architecting Secure Solutions) attributes to the available IBM Tivoli solutions in the Identity and Credential Management league and provide a variety of customer scenarios that all require some security-related data synchronization.

20.1 Identity data management aspects of MASS

There are many aspects to building a security architecture. The goals of MASS are to use existing security paradigms, integrate with other information technology architectures and work with today's technologies. MASS analyzes the 11 functional classes and the 130 component criteria of Common Criteria. Common Criteria^[1] is recognized internationally as supported *best practices* and is intended as a standard for evaluation of security functionality in products. MASS has effectively taken these breakdowns and has established five functional categories (see Figure 20-1) 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.

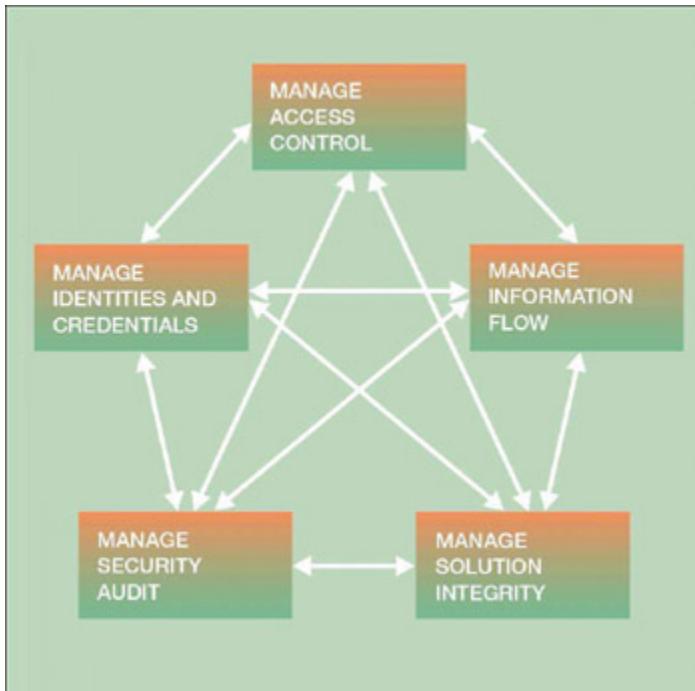


Figure 20-1: MASS subsystem overview

More information about MASS can be found in [Chapter 2, “Method for Architecting Secure Solutions”](#) on [page 15](#).

[1]For reference, visit <http://csrc.nist.gov/cc/>

20.2 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 have described several applications that rely on directory data for operations (Access Manager, Privacy Manager, and 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.

20.3 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.

20.4 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 20-1](#) on [page 504](#) 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 20-1: 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	

Process/function	Tivoli Identity Manager	Tivoli Directory Integrator
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
Prioritization of service		
Automated policy enforcement/workflow	X	
Policy enforcement	X	X
Policy administration	X	
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.

20.5 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.

20.5.1 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.

Gartner estimates that 70% of all large firms will deploy a metadirectory solution by 2005. 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 can not 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 20-2 on page 507](#) 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.

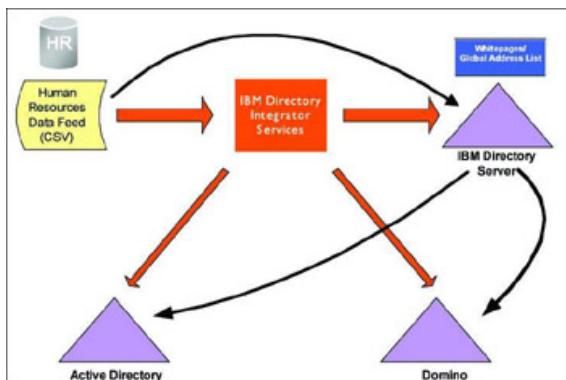


Figure 20-2: Metadirectory services with Directory Integrator

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. 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.

20.5.2 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 20-3 on page 509](#) illustrates two different use cases of Directory Integrator for identity data management in Identity Manager.

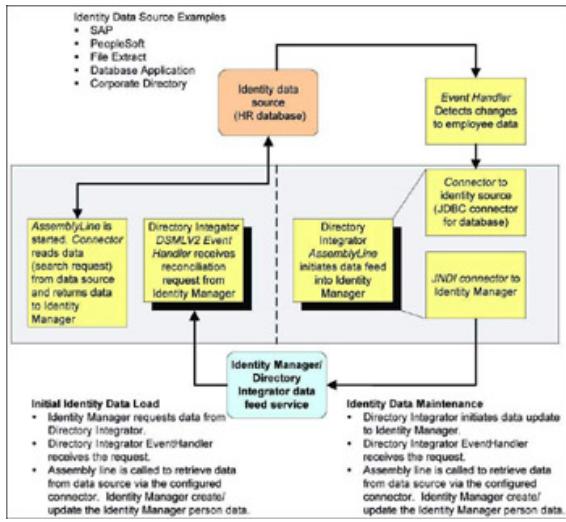


Figure 20-3: Accelerating Identity Manager deployments

Initial data load

In Identity Manager 4.5, integration with Directory Integrator is build into the Identity Manager server and into Directory Integrator 5.1.2. One specific feature 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 the quickest way to do a bulk data load of user information into Identity Manager.

Directory Integrator 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 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. An example might be assigning a role or alias.

5. Data is returned to Identity Manager 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 set the Identity Manager attribute erPersonStatus to 1, 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 frequent) basis.

Using the LDAP changelog connector, changes to that branch of the schema for a person's account can be detected (for example, Ou=0, ou=accounts, erglobalid=00000000000000000000, ou=companyname, dc=com).

If there is a change to the e-mail address, Directory Integrator can be used to update the person information branch with the changes from the account branch (for example, Ou=0, ou=people, erglobalid=00000000000000000000, ou=companyname, dc=com).

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.

20.5.3 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 20-4](#) depicts the solution overview.

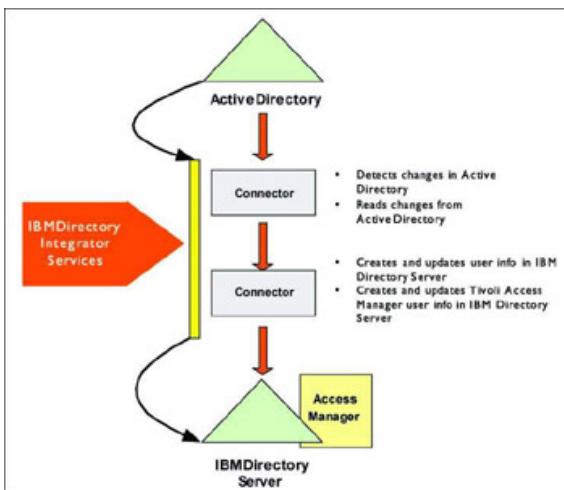


Figure 20-4: Multiple directories and Tivoli Access Manager

The data flow is 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 using the uSNChanged mechanism. 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 ePerson object class that is used by Tivoli Access Manager.
 - b. Use the Tivoli Access Manager Java API to create and modify the entry information that is used by 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.

20.5.4 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. A Gartner study indicates that for some companies, in 80% of the calls significant 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, SunOne Directory Server, and IBM Lotus Domino HTTP passwords. The Password Synchronizer for Windows intercepts changes in user accounts on Windows NT, Windows 2000, and Windows XP operating systems.

Another unique 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 and Tivoli Access Manager. In Identity Manager 4.5, the server that 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 an RACF 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 RACF 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 20-5](#) on [page 515](#) provides an overview of the solution.

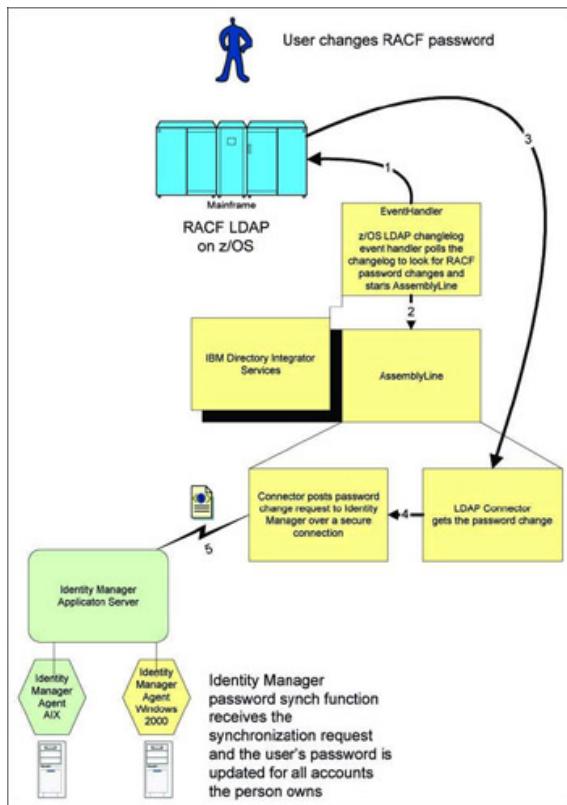


Figure 20-5: Password synchronization services

The event flow is:

1. The user changes a password in RACF. This change is recorded in the z/OS LDAP changelog, and the password is encrypted and stored in a reserved attribute in LDAP. The Directory Integrator EventHandler for z/OS LDAP detects the password change.
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, AIX, Windows NT, 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.

20.5.5 E-mail migration services

IBM Lotus Workplace Messaging™ provides a Web-based mail application. Workplace Messaging is designed to integrate with an existing corporate infrastructure and use the corporate LDAP directory for automatic user account creation, deletion, authentication, address-resolution, and mail routing.

To enable customers to get up and running on Lotus Workplace Messaging quickly, IBM Directory Integrator can be used to aid in the migration of existing users from Microsoft Exchange or Lotus Domino and is bundled with the Workplace Messaging offering.

To migrate the mail from legacy mail systems, an XML file must be created that contains all of the information needed to facilitate the migration of the mail content and contacts from a specified legacy mail account to a specified Workplace Messaging mail account. You generate migration requests outside Lotus Workplace Messaging using the IBM Tivoli Directory Integrator and the migration AssemblyLine, a configuration XML script interpreted by Directory Integrator. The XML script reads a set of properties to generate the migration request. Each mail migration request specifies whether mail, contacts, or both are to be migrated. For migration to work, both mail and contact information must be on the legacy mail server.

Figure 20-6 shows how migration requests are generated.

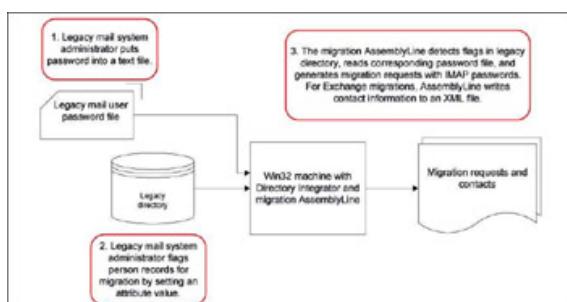


Figure 20-6: E-mail migration services

When a user is flagged for migration, Directory Integrator builds a request to migrate the mail file, contact list, or both. The mail file part of the request contains information so that the migration process in Lotus Workplace can later make an IMAP connection and pull over the user's mail file.

After the migration XML documents are created, you use the **migrate** command to process the migration requests so that the mail content and contacts are imported into the target mail account in Workplace Messaging. IMAP access to a user's mail file requires the user login name and password. These credentials must be provided as part of the migration request in a form that is accessible to the **migrate** command.

For this solution, two sets of Directory Integrator AssemblyLines are provided with the Lotus Workplace Messaging Product: *Migrate* and *Migrate and Coexist*.

The Migrate AssemblyLines make user migration requests and pull some data based on the contents of the legacy directory.

The Migrate and Coexist AssemblyLines can make user migration requests and pull some data based on what is in the legacy directory. They also can *provision* (create) a new user account in the Lotus Workplace LDAP Directory for the migrated user. They also sync the entire Legacy Directory (Domino, Exchange 5.5, Exchange 2000) into the Lotus Workplace LDAP.

This example shows how Directory Integrator can be used as a utility program to facilitate the extraction of data to be used for a variety of data exchange and synchronization purposes.

20.5.6 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-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.

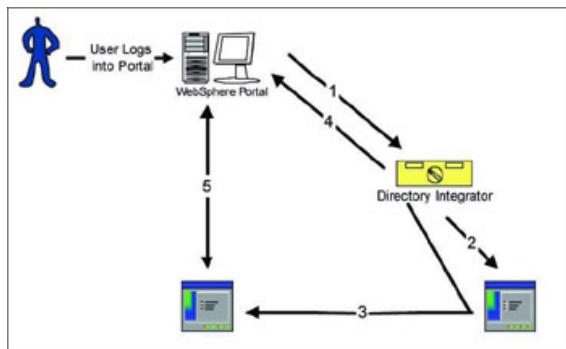


Figure 20-7: Enabling Web portals

Conceptually, the solution flow depicted in [Figure 20-7](#) works as so:

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.

For more detailed information about this solution, read the document at:

http://www.software.ibm.com/wsdd/library/techarticles/0306_lawrence/lawrence.html

20.6 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.

Part 4: Managing a Security Audit

Chapter List

[Chapter 21:](#) Risk Manager Topology and Infrastructure

[Chapter 22:](#) Building a Centralized Security Audit Subsystem

[Chapter 23:](#) Extending the Centralized Security Audit Subsystem

In this part, 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 Risk Manager. Risk 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.

Chapter 21: Risk Manager Topology and Infrastructure

Before going into any specific customer scenario, we want to focus on a general example layout of an enterprise e-business IT infrastructure and how Tivoli Risk Manager contributes to it. Because a lot of the components mentioned in this overview will be found in almost every implementation, the integration of these components into a centralized enterprise risk management implementation will be very similar to ours.

Tivoli Risk Manager builds on the Tivoli Framework infrastructure, so we outline the general aspects of integration. A more detailed Tivoli Framework overview with the necessary components is found in Tivoli Framework architecture. Based on the Tivoli Framework foundation, we briefly describe all of the available Tivoli Risk Manager elements and how they fit together. Detailed information about each of these is found in the relative product documentation.

21.1 Across boundaries

Information Technology has became vastly important, but the consistent increase of security threats calls for an integrated, multivendor approach to security management. New forms of security attacks constantly emerge, and organizations must address the business risks arising from these security issues:

Viruses	A small piece of computer code that piggybacks to other legitimate computer software.
Worms	Software codes similar to viruses that make use of security loopholes in operating systems and spread from one system to another via networks.
Trojan horse	A program that fakes its purpose and claims to be a useful software utility.
Port scanning	A method adopted to scan computer systems in the network for open ports.
Denial of Service attacks	These render a server or site useless by sending large amounts of garbage data and bogus requests.
Remote Administration	Back doors usually refer to remote administration programs.
Sniffers	Packet-capturing programs that are used to obtain logon IDs, passwords, and other confidential information.
Spoofing	Hiding one's own identity and posing as another to gain trust.

Intrusion

Unauthorized access to hosts, Web servers, private networks or databases.

There are a number of ways to configure the IT infrastructure for an enterprise, depending on the size of the organization and on what you are trying to achieve. An enterprise must deploy at least a few of these products to secure the IT infrastructure solutions:

- Antivirus software
- Intrusion detection systems
- Firewalls
- Public key cryptography
- Spam filters
- Security compliance manager
- Health check software for system security compliance
- Access management software
- Identity management software
- Privacy management software
- Database management software

We now look at the IT infrastructure and the network topology typically deployed in a sample e-business environment [Figure 21-1](#) depicts a network topology with multiple zones:

1. Non-secure network

A public network that uses the Internet protocol and the public telecommunication system to share business information or part of an operation with customers, suppliers, vendors, partners, or other businesses.

2. DMZ

The demilitarized zone (DMZ) is a computer host or small network that acts as a *neutral zone* between a company's private network and the outside public network. It prevents outside users from gaining direct access to a server that has company data. It is an optional and more secure approach to a firewall and effectively acts as a proxy server as well. Users of the public network outside the company will only access resources in the DMZ.

3. Secure network

The secure network, or intranet, is a private network within an enterprise. Typically, an intranet includes connections through one or more gateway computers to the Internet. The secure network can have multiple parts:

- The production zone, which holds databases, Web servers, and Web application servers, is a *restricted zone*.
- Intranet users are located in a *controlled zone*.
- The management zone, which holds management servers such as systems management, security management, and storage management, is called a *secured zone*.

4. Extranet

This term is only a *state of mind* in which the Internet is perceived as a way to do business and connect with other companies, such as subcontractors or business partners. The intention is to establish secure connections from one company to the other by using the Internet as the network provider.

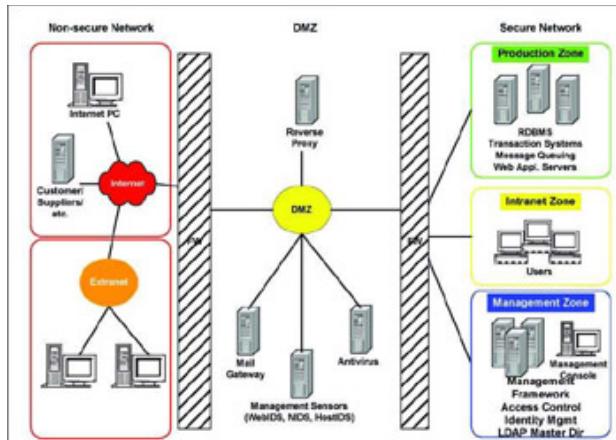


Figure 21-1: e-business network topology

For a general overview of network zones refer to [Chapter 3, “IT infrastructure topologies and components”](#) on [page 49](#).

21.2 Overview of components

In [Figure 21-1](#) on [page 525](#), we show a complex e-business scenario, part of that might be found in almost every enterprise environment. This section briefly discusses the components involved in this infrastructure.

21.2.1 Web server environment

The Web server environment includes different kinds of Web servers, Web application servers, and corresponding management and security technologies for these servers.

Web servers

A Web server provides informational content in the form of Web pages to Web browsers. Web servers may contain pages about a company, product offerings, technical information, or other static content. Web servers also offer some ways of providing dynamic content and scripting capabilities. Due to security and management reasons, these areas are delegated more and more to specialized Web application servers.

Web application servers

Web application servers provide the business logic for an application program in a distributed network environment. These applications are mainly based on Java servlet or Enterprise Java Beans (EJB) technology. Web application servers provide access to valuable business data through browser-based access models. They should not be placed in the DMZ, but rather in a *restricted* zone.

Access Control Management

IBM Tivoli Access Manager is a robust access control management tool for e-business and distributed applications that takes care of authentication and authorization for accessing enterprise data. It includes:

- HTTP Reverse Proxy (WebSEAL)
- Access control for Web applications
- Authorization service and application programming interface (API) for legacy and distributed application integration based on C++ and Java
- Access control for MQSeries-based applications
- Access control for UNIX-based resources and applications
- Highly available and performing management server infrastructure

21.2.2 Back-end data

The back-end data reflects the most valuable assets a company owns, not only in IT terms. This is always protected at highest priority as far as security architecture design is concerned.

Database systems

A database holds information that is organized, so that its contents can be easily accessed, managed, and updated. Databases contain aggregations of data records or files, such as sales transactions, product catalogs and inventories, and customer profiles. Typically, a database manager provides users the capabilities of controlling read/write access, specifying report generation, and

analyzing usage. Databases and database managers are prevalent in large mainframe systems, and can be found in smaller distributed-workstation and mid-range systems. Controlling access to these systems and protecting them against fraudulent action is most important.

Transaction systems

A transaction usually means a sequence of information exchange and related work (such as database updating) that is treated as a unit for the purposes of satisfying a request and for ensuring database integrity. For a transaction to be completed and database changes to made permanent, a transaction has to be completed in its entirety.

Message queueing

Message queueing is a method by which processes asynchronously exchange or pass data using an interface to a system-managed queue of messages. This queue is created by one process and used by multiple other processes that read and write messages to the queue.

Message queueing connects many commercial systems and works independently of network disruptions, so that important data is always delivered.

Management framework

Management framework consists of systems help for systems management, network management, security management, and so on.

21.2.3 Firewall

A firewall is a program, located at a network gateway, that protects the resources of a private network from outside

users. Security policies implemented at the network level, host level, and application level allow access only to authorized users, applications, and systems, depending on policies defined.

An enterprise with an intranet that allows its workers access to the wider 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 toward its destination. A firewall also includes or works with a proxy server that makes network requests on behalf of workstation users.

A firewall is often installed in a specially designated computer separate from the rest of the network so that no incoming request has direct access to private network resources.

Any abnormal attempt to access network resources through firewall devices has to 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 the security management tools, access to the threat events on the network.

21.2.4 Router

A router is a device that determines the next network point to which a packet should be forwarded toward its destination. The router is connected to at least two networks, and it decides which way to send each information packet based on its current understanding of the state of its connecting networks. A router is located at any gateway (where one network meets another), and is often included as part of a network switch.

A router may create or maintain a table of the available routes and their conditions and use this information along with distance and cost algorithms to determine the best route for a given packet. Typically, a packet may travel through a number of network points with routers before arriving at its destination.

21.2.5 Intrusion detection system

An intrusion detection system (IDS) is a type of security management system for computers (Host IDS), Web servers (Web IDS), and networks (Network IDS). An IDS gathers and analyzes information from various areas 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). The IDS uses *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
- Analyzing system configurations and vulnerabilities
- Assessing system and file integrity
- Recognizing typical patterns of attacks
- Analyzing abnormal activity patterns
- Tracking user policy violations

Intrusion detection systems are being developed in response to the increasing number of attacks on major sites

and networks, including those of the Pentagon, the White House, NATO, and the U.S. Department of Defense. The safeguarding of security is becoming increasingly difficult, because the possible technologies of attack are becoming ever more sophisticated; at the same time, less technical ability is required for the novice attacker, because proven past methods are easily accessed through the Web with the tools to help.

Web IDS

Web Server intrusion detection has been introduced with IBM Tivoli Risk Manager. It uses the actual access log files generated by a Web server to perform the analysis that detects the Web server attacks. Web IDS, which is deployed on each Web server, monitors the log in real time using a knowledge-based approach to detect malicious access attempts.

Most companies use a multitude of IDSs on a variety of systems in order to gain as much information as possible about malicious access attempts. It is very challenging to stay focused with the amount of alerts and false positives coming in from all IDSs on a network.

21.2.6 Antivirus

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. The servers can check for and download updated virus signatures in order to upload them to the agents running on user desktops to check for virus infection to the systems.

21.2.7 Mail infrastructure

The e-mail system is the most vulnerable for spreading viruses. E-mail is one of the protocols included with the Transmission Control Protocol/Internet Protocol (TCP/IP). The most popular protocol for sending e-mail is Simple Mail Transfer Protocol (SMTP); for receiving, it is Post Office Protocol Version 3 (POP3).

21.3 Tivoli Framework architecture

Tivoli Risk Manager adds enterprise risk management to the comprehensive list of enterprise management solutions and third-party applications based on the Tivoli Management Framework. Risk Manager V4.2 works with IBM Tivoli Management Framework V4.1 with required patches and a supported database. The Tivoli management architecture manages thousands of machines (endpoints) from a single Tivoli Management Region (TMR) server. Servers interconnect to manage large, multiple-domain networks.

The Tivoli Management Framework is based on a three-tier architecture. [Figure 21-2 on page 531](#) shows the four basic components: TMR server, Tivoli Management desktop, Tivoli Management agent, and Tivoli Management gateway.

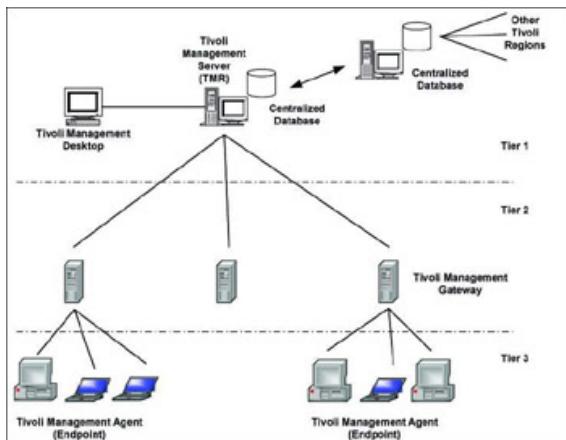


Figure 21-2: Tivoli management architecture

The Tivoli Management Framework provides a set of common services and features that are used by other Tivoli applications installed on top of the Tivoli Management Framework. These services include, but are not limited to:

- A task library, which is a mechanism for creating tasks and executing them on multiple Tivoli resources. A task can be an application, a shell script, or a batch command file.
- A scheduler that enables you to schedule all Tivoli operations, including the execution of tasks created in the Tivoli task library. Scheduler follows the real-time clock on the Tivoli Management Region server to execute the scheduled activities.

- An RDBMS Interface Module (RIM) that enables some Tivoli applications to write application-specific information to relational databases. This module has authenticated access, as it uses defined passwords for connection to the RDBMS.
- A query facility that enables you to search and retrieve information from a relational database. There are some pre-defined queries as well as user-defined queries written to fetch the data.

21.3.1 Tivoli Management Region server

The Tivoli Management Region server (TMR server) includes the libraries, binaries, data files, and graphical user interface (GUI) needed to install and manage your Tivoli environment. The TMR server maintains an internal database that includes information about the managed objects, such as workstations, databases, and applications, and it coordinates all communication with Tivoli-managed nodes. In addition, the endpoint manager is part of the TMR server and is responsible for endpoints. The server also performs all authentication and verification necessary to ensure the security of Tivoli data. Tivoli desktop is used to configure TMR server functionaries.

21.3.2 Tivoli Management Desktop

The Tivoli Management Desktop enables the administrator to access the Tivoli management applications from any location in the network, with any desktop that has Tivoli Desktop agent installed.

When you install the TMR server on a UNIX system, the Tivoli Management Desktop is automatically installed. When you install the TME® server on a Windows NT or Windows 2000 system, you must install it separately.

21.3.3 Managed node

A Tivoli *managed node* runs the same software that runs on a TMR server. Managed nodes maintain their own databases, which are accessed by the TMR server. When managed nodes communicate directly with other managed nodes, they perform the same communication or security operations performed by the TMR server.

The difference between the TMR server and the managed node is that the TMR server database is global to the entire TMR, including all managed nodes. In contrast, the managed node database is local to the particular managed node. For management of the computer system that hosts the managed node, install an endpoint on that managed node.

21.3.4 Tivoli endpoint

An endpoint is a PC or UNIX workstation running the Tivoli Management Agent. It enables one-touch management by quickly detecting changes within the environment and bringing them under Tivoli management. Each event source that is integrated into the management framework must be configured as an endpoint. Endpoints provide the framework services to Tivoli applications that run on the workstations or desktops. Adapters (TME Adapters), which are described in 21.4.1, “Risk Manager Adapters” on [page 540](#), use the endpoint to communicate alerts to the Tivoli event management platform.

21.3.5 Tivoli Firewall toolkit

Where events are generated on one side of a firewall and have to be sent to a TEC server on another side, Tivoli firewall toolkit should be used to save opening the firewall too much. However, as it is useful only for data being transported via oserv, it is used only for Risk Manager events when they are on their final hop in the chain to the TEC Console server.

21.3.6 Gateway

A *gateway* controls all communication with and operations on Tivoli Endpoints. A single gateway supports communication with thousands of endpoints. A gateway launches methods on an endpoint and runs methods on the endpoint’s behalf.

A gateway is created on an existing managed node. This managed node provides access to the endpoint methods and provides the communication with the TMR server that the endpoints occasionally require.

21.3.7 Tivoli Management Desktop

The Tivoli Management Desktop is a graphical user interface (GUI) that presents an administrator’s view of the Tivoli environment. From the Tivoli management desktop, resources and tasks are easily accessed across the enterprise.

The Desktop in [Figure 21-3](#) on [page 534](#) depicts a variety of resources, including:

- Administrators
- EndpointManager

- Policy regions
- Schedulers
- Generic collections
- Application-specific resources



Figure 21-3: Tivoli Management Desktop

21.3.8 Policies and policy regions

Policies and policy regions play a fundamental role in Tivoli architecture. In the Tivoli environment, a policy is a set of rules applied to managed resources. It enables the administrator to control the default values of newly created resources (default policy) and maintain the guidelines when administrators modify or operate on resources (validation policy).

A specific rule in a policy is a policy method. A default policy method supplies a constant value or runs a shell script or a program that generates a value, whereas a validation policy method typically runs a program or shell script to verify values supplied by the administrator. Administrators define and maintain policies.

Policy regions are containers for managed resources that use the same set of policies (for example, a collection of desktops within a given administrative domain). Collectively, these desktops are administered using permissions or roles that are assigned to administrators on the basis of policy regions. Policy regions help organize the managed resources in the desktop and define and limit administrator access to these resources.

21.3.9 Centrally managed policies

Tivoli Risk Manager leverages policy regions, which enable security management staff to centrally manage policies and determine who is allowed to support which targets and in what manner. This is done on the basis of skills (for example, antivirus administrators) or on the basis of geographical location (for example, New York branch analysts).

Tivoli Risk Manager offers policy-based delegation of authority, and IT management groups target into policy regions. If a simple marketing department policy region contains four PCs, the antivirus administrator supporting the marketing department has authorization to access this policy region. A senior administrator sets up the policy region and delegates the support of the marketing department to a junior administrator. In this case, the defined policy may require antivirus signatures to be updated to all desktops every week. The administrator uses a virus signature profile to implement the organization's antivirus update policy.

The policy regions in Tivoli Risk Manager provide granular access control and implement enterprise risk management in large organizations. Centrally managing policy makes the product scalable, because the complexity of management does not increase exponentially as the size of the network increases. Administrators are assigned roles on a region-by-region basis. Roles help restrict the level of operations that an administrator performs.

21.3.10 Tivoli Enterprise Console

Tivoli Enterprise Console (TEC) is a robust platform for centralized event correlation engine across the enterprise. Events generated from distributed event sources are stored in a relational database and managed by the Tivoli enterprise event console. System administrators use the Tivoli desktop console to manage events from applications, such as Tivoli Distributed Monitoring and Tivoli Security Manager, and to manage events generated from third-party applications.

Prerequisite for Risk Manager V4.2 is Tivoli Enterprise Console Version 3.9. Risk Manager Web applications also require WebSphere Application Server 5.0.1 (used by TEC) in order to explore event details for incident events, and the events causing the incidents.

Tivoli Enterprise Console collects and integrates disparate management information into a common model for event processing and provides a central operations view. Types of event processing include event

correlation, filtering, dropping duplicates, prioritizing, consolidating, closing self-correcting events, escalating events, and forwarding events. Tivoli Enterprise Console handles events from applications, databases, and systems and network devices.

When events are defined, Tivoli Enterprise Console rules correlate events and define automated actions. Correlation automatically closes events related to resolved problems. An automated response capability enables problem resolution with no user intervention.

Security analysts use Tivoli Risk Manager to quickly troubleshoot problems logged in to Tivoli Enterprise Console, such as monitoring open events, generating trouble tickets for problem resolution, or responding to serious events with actions. The new Tivoli Enterprise Console panel in [Figure 21-4](#) depicts an overview of Risk Manager events.

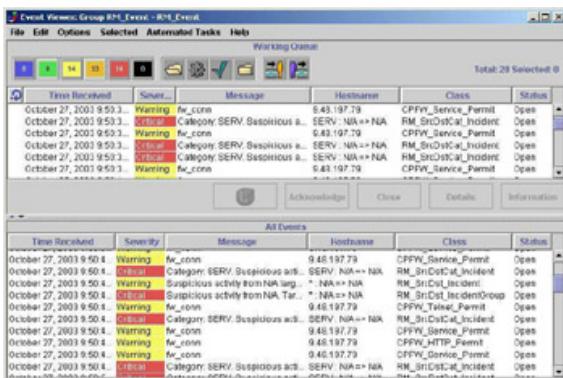


Figure 21-4: Enterprise console: all Risk Manager events

More details about each individual event are available, as shown in [Figure 21-5 on page 537](#). A request for information about an event initiates the Web console and, when the user signs in, he gets access to the Event Details and Advisor options. The event details let you browse through the events-causing incident.

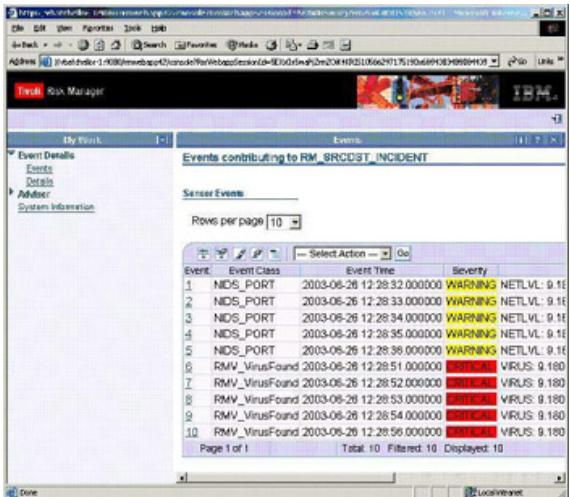


Figure 21-5: Risk Manager incident details

The advisor enables options to browse through the system information if a Tivoli Inventory module is installed, and it offers an option to configure alert mail in the event of an incident to defined users.

Refer to the Risk Manager product documentation for more information about the Web access configuration.

Tivoli Enterprise Console Version 3.9 adds significant performance enhancements with hundreds of events per second throughput and high-capacity event throttling. It enables extensive flexibility and scalability in large environments, and it offers high-speed, logic-based reasoning to centrally control and manage the intelligence with drag-and-drop interfaces.

There are three main components in the Tivoli Enterprise Console:

- Event Sources

These applications gather information about resources throughout the IT environment. They typically run on any machine where availability is of concern. After the information has been gathered, it is converted into an event and forwarded to the event server.

- Event Server

The event server is a group of cooperating processes that run on a single managed node within a TMR. These processes cooperate with each other to receive events from various sources, perform various levels of processing on received events, and forward the events to the corresponding event consoles. The event server is the heart of

TEC, as it provides the Prolog engine for applying rules to received events.

- Event Console

This is the user interface that an administrator uses to interact with the Enterprise Console. Each instance of an event console is configured to display logical groupings of events.

The administrator performs actions on events displayed on the console, such as viewing the detail. Any changes made to an event from other consoles or the event server is reflected on each console that has the event in its view.

For more information about the TEC, refer to TEC 3.9 product documentation.

21.4 Risk Manager architecture

There is drastic design change with the new version of IBM Tivoli Risk Manager 4.2 as compared to Version 3.8. Risk Manager 4.2 requires the Tivoli Enterprise Console 3.9 and the Tivoli Management Framework 4.1. A Risk Manager [Agent](#) is available with the *Event Monitor* that provides the TEC Logfile Adapter capability. Correlation processing is off-loaded onto a *Distributed Correlation Engine* (DCE). The event archive database is loaded and accessed separately from TEC, sparing TEC from this resource-consuming task.

It became apparent that one of the major differences between security management and other kinds of systems management disciplines was not being adequately addressed. A typical systems management environment may generate thousands of events per day, and you see similar numbers in a security management environment—until a real-life attack happens. At that point, hundreds of thousands of security threat events can be generated per hour. This was far more than the Enterprise Console was designed to handle. A new approach was required that retained the power of the existing adapter, correlation engine, and reporting model, and could satisfy the following needs:

- Much higher thresholds for event throughput overall (scalability)
- More options to process events throughout the network (flexibility)
- Less dependence on the Tivoli framework for transport while maintaining or improving security of the event transport
- Easier installation and setup

At the same time, a focus was needed on providing more timely updates to adapters and expanding platform support for all of the key components. Finally, a different and more universal reporting mechanism was required.

These needs led directly to a more distributed architecture, in Risk Manager 4.2, that changed the product in many ways. The new Risk Manager architecture is shown in [Figure 21-6](#).

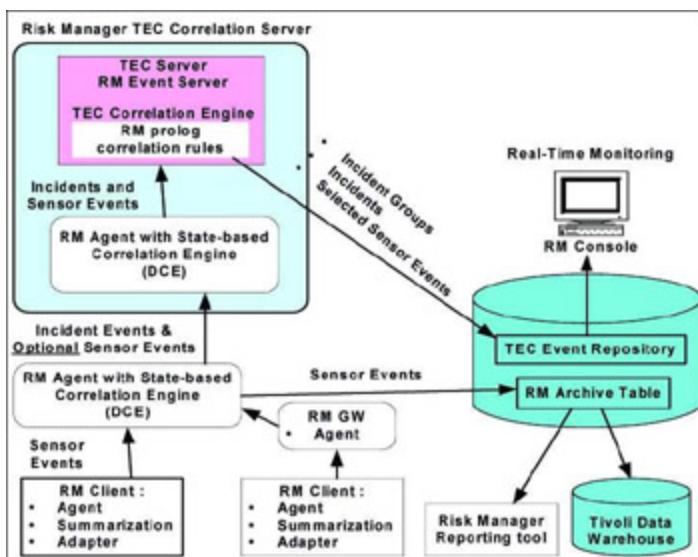


Figure 21-6: Risk Manager Architecture

There are four key components of Risk Manager that are based on the Risk Manager Agent. This means that they have the agent code and configuration files in common. Wherever we install an agent, no matter what values we use during the installation, it is always possible to manually change one agent to behave as any of these four components (or sometimes, more than one at a time, such as a client and a gateway configured on the same [Agent](#)).

These four key components for Risk Manager are:

- Event Server

- Distributed Correlation Engine (DCE)
- Gateway
- Client

A second key concept is a subcomponent of Risk Manager called the Zurich Correlation Engine (ZCE). This is a state-based correlation engine that was developed by the IBM Zurich Research Labs. It is a specialized pattern-matching engine that is utilized by the Agent running on the Client, the DCE, and the Event Server. It is additional to the well-known prolog-based correlation engine that runs on the TEC. Unlike in the TEC environment, the end user and even the development staff in an organization using Risk Manager will never interface directly with the ZCE. Risk Manager has wrapped Java programs around the ZCE, and it is the Risk Manager programs that the end user interfaces with, usually by altering XML configuration files.

Other components of the Risk Manager architecture are:

- Risk Manager Adapter
- Risk Manager Archive Database
- Tivoli Enterprise Console
- Reporting functions

In addition, parts of the overall Risk Manager solution require the following optional products:

- Crystal Reports (prepackaged reports and a viewer are supplied with the Risk Manager product)
- Tivoli Inventory

Attention It is possible to customize Crystal Reports or add reports of your own. To do this you will need to purchase the Crystal Reports Designer product. This is not supplied with Risk Manager. The 22 pre-canned reports are viewed out-of-the-box.

Everything that is needed for a basic deployment from event capture to reporting comes packaged with the product. Adapters for all of the supported devices are available through a support Web site:

<http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliRiskManager.html>

21.4.1 Risk Manager Adapters

In most respects Risk Manager Adapters perform tasks identical to any other TEC adapter you may have dealt with. They monitor event sources and convert the events into a standardized format that will be securely forwarded to the Tivoli Enterprise Console Event Server that uses the Risk Manager Agent for further state-based correlation. This has changed slightly with the advent of the Event Monitor in Risk Manager 4.2, but for those choosing not to use the Event Monitor, the Risk Manager Adapter will be very familiar if you are used to TEC. The Event Monitor is introduced in 21.4.6, “Event Monitor” on [page 543](#). The job of the adapter is to:

- Capture events, such as:
 - Firewall log file alerts
 - Network and host-based intrusion detection systems

- Syslog messages from UNIX
 - Event logs generated from Microsoft Windows
 - Antivirus alerts generated by desktop virus software
- Recognize the events to match the format files
 - Transform each individual event into a standard format that the TEC would recognize for further processing
 - Apply any user-defined filters
 - Forward the event on to the next stage of processing

Deployment of adapters is time consuming and complex in a large environment, just as it is with standard TEC adapters. Tivoli tools such as the Adapter Configuration Facility can help you configure your adapters using XML file configurations.

TME and non-TME adapters

This refers to the transport being used when the adapter sends events onward. A TME adapter sits on an endpoint and uses Framework services to transmit events (using oserv, the protocol associated with the Framework). A non-TME adapter does not have these facilities available to it and so uses TCP/IP. In the case of Risk Manager adapters, Secure Socket Layer (SSL) may also be used.

Most Risk Manager adapters support either of these transport mechanisms.

21.4.2 The Risk Manager Client

This is the first of the components that is a manifestation of the Risk Manager Agent. The client receives events from one or more adapters. At the heart of the client is a sub-component known as the summarization engine. The job of the summarization engine is to keep the network from flooding with events during a full-scale attack. Instead of sending 100 identical events (apart from their timestamps, of course), the Client sends one event marked as a representative of the 100 events (with repeat count 100), while the original events are discarded.

What summarization is done and the thresholds for summarization are fully configurable. The Client does not attempt to do any correlation as this is the next stage in processing and it uses its own specialized agent.

The client is the first place in the event flow where we meet a Risk Manager Agent. It is installed via an Install Shield Multi Platform (ISMP) GUI, which carries out most of the configuration work needed to have a working client. The job of the Client is to receive events sent from one or more adapters and to invoke the ZCE, which is configured to work as a summarization engine.

21.4.3 Risk Manager DCE

The Risk Manager Distributed Correlation Engine (DCE), is key to the distributed architecture puzzle. This component receives events from one or more clients (though it is quite capable of handling events directly from adapters and, in certain circumstances, this may indeed be a suitable choice) and provides the main correlation that is performed to produce the *incident* that will be seen on the Tivoli Enterprise Console. This is performed by the state-based ZCE. Out of the box, it implements the correlation algorithm that has been used since the early days of Risk Manager,

but a sophisticated user fine-tunes these to suit his own environment.

Following the processing of the sensor events received, the DCE sends the generated incident events up the chain for processing at TEC. At this point, the user chooses whether to send any of the original sensor events onward to TEC with the incidents. (This is safe to do even if another DCE is further up the chain, as events are marked as having been correlated after they have been through a DCE engine).

21.4.4 Risk Manager Gateway

At some point in the process of transferring events across the network, the data stream may have to traverse a firewall. In the case of standard TEC adapters, this has been an area of concern, it needs to open numerous ports on the firewall. The development of the Tivoli Firewall Toolbox provided a solution for TEC adapters. However, as non-TME Risk Manager adapters do not use the Tivoli Framework for transport, it is necessary to provide an alternative mechanism to do the same thing. The Gateway does not process events at all, but merely receives them at one defined port and passes them on to the next phase of processing using another defined port. It sends and receives via either TCP/IP or SSL. In this way, by having one Gateway in front of the firewall, it uses a single connection to forward information through firewall environments. This significantly reduces the possibility of a hacker compromising the firewall itself. The Gateway is used in our customer scenario in 22.2.3, “Integration of Risk Manager” on [page 569](#).

21.4.5 Risk Manager Event Server

Often referred to as simply the Event Server, this component is installed on the same physical system as the

Tivoli Enterprise Console (TEC) server. This component obviously has a central role to play in the overall processing of security events. It is comprised of a Risk Manager Agent with a Correlation Engine enabled (to provide any required top-level correlation for the whole network) and is configured to pass on any incident events received or generated to the TEC server. It is also the last point in the overall layout where sensor events are independently siphoned off to the Risk Manager Archive database. The Event Server is mostly used as a central gathering point for incident events coming in from all over the network.

21.4.6 Event Monitor

The standard TEC logfile adapter has some known limitations: You cannot have more than one of them active on a Windows system at one time, and TEC adapters only monitor one log file at a time. This is a limitation in a complex Tivoli environment where, for example, you may want to install Tivoli Monitoring as well as Risk Manager. The Event Monitor addresses this, providing a built-in mechanism to monitor multiple log files as well as several excellent performance options.

21.4.7 Risk Manager Archive database

This is a database of all Risk Manager events that have been sent to the Archive database at various stages of processing. This database is very important for the Web-based applications and for the Reporting functions. All of the Agent-based components send events to the Risk Manager Archive database if they are properly configured, so the end user has a lot of choice about event flow and how much to send through any particular point in the network.

21.4.8 Reporting

Regardless of the mechanism by which reports are generated, they represent an important part of the overall functionality of Risk Manager. Use of reporting tools (with data fed from the Risk Manager to Archive database) point out long-term patterns of attack or suspicious activity that will be lost in the day to day operation and management of the security environment. Correct use of reporting tools enables a business to work proactively to protect its assets, rather than always having to be in react mode.

Crystal Reports

Near-term reports are provided for this version of Risk Manager using Crystal Reports. Crystal Reports use archive table (*arc*). Creation of this table is not part of the standard Tivoli Enterprise Console installation. It is a single table that is created manually, and the Tivoli Risk Manager Agent Database Pusher is used to update these tables by DCE. It must be large enough to hold data for a reporting period covered by your requirements for online reporting by Crystal Reports. Refer to the *Risk Manager 4.2 Installation Guide* and the *Risk Manager 4.2 Administration Guide* for further information about installation of crystal reports and related configuration.

Tivoli Data Warehouse for Risk Manager

This product is available as Tivoli Data Warehouse enablement pack. It provides data storage and retrieval techniques that enable detailed long-term reporting. As it is used by many Tivoli products, it will be a central point for data collection and correlation of Risk Manager information with information from other sources. In addition to all the above, a relational database manager is required for storage

of the TEC and archive databases, and the relevant level of Java must be installed because much of the Risk Manager product is written in this language. For the Web-based applications, one of the supported Web servers is also required. Risk Manager TEDW enablement pack will load the Extract, Transform, and Load (ETL) component to load the data from event repository to TEDW database tables and from TEDW database tables to Data Mart.

This is the strategic repository for Tivoli data. Tivoli data sources using TEDW include products from the Tivoli Monitoring family, TEC, Risk Manager, Tivoli Business Systems Manager (TBSM), Configuration Manager, and Access Manager (via its integration with Risk Manager). Products that extract reports from TEDW include Service Level Advisor.

21.4.9 Event processing overview

Whenever a security device (or *sensor*) detects an attack or anomaly, one usual response is to try to alert an operator in some way. This is usually done by producing alerts of some kind. These messages may be stored in a log file, displayed on a console, or sent by some means to another program that intercepts and interprets them. Many security devices come with a console of their own, where the alerts are displayed for the systems operator. Let us take a closer look at the event flow when Risk Manager is involved:

1. An attack happens and a security sensor generates alerts and writes them into a local log file.
2. The alerts are picked up from where they are stored or intercepted by an adapter.
3. The adapter standardizes the events, selects those that are important, and sends them upstream.

4. A client receives the events and, if there are a large number of duplicates, summarizes them and forwards the summarized event to the next stage of processing.
5. The DCE picks up the events from the clients and begins to look for higher-level patterns. It produces an *incident* based on correlation rules and sends the events to the next stage of processing. At this stage, any sensor events that have come this far and the summarization events may be sent to the Risk Manager Archive database or else they will be sent on for further processing. Note that after an event has been through a correlation engine it is marked as *correlated*, and it will not be included in any further DCE-based correlation.
6. The events arrive at the TEC. If a number of incidents with similar attributes are received, the TEC correlation engine creates Incident Groups to refer to them. Events, incidents, and incident groups are displayed on the console.
7. Based on rules defined on TEC, execute any of following tasks:
 - a. Send an alert mail to an administrator.
 - b. Inhibit a connection to or from a specific IP address on the firewall.
 - c. Inhibit and close any existing connections to or from a specific IP address on the firewall.
 - d. Cancel a previously enabled action on the firewall.

- e. Cancel all previously enabled actions on the firewall.
 - f. Terminate a user process on a server.
 - g. Suspend a user account on a server.
 - h. Scan or delete a virus on a desktop.
8. Reports can be run against the events in the Archive database to look for longer-term patterns.
-

21.5 The benefits of enterprise risk management

Risks are pervasive throughout the enterprise. Enterprise risk management lowers the overall risk to the enterprise by leveraging security intelligence across many different security products. There are four major management disciplines:

- Firewall management
- Intrusion detection
- Risk assessment
- Virus management

21.5.1 Firewall management

Businesses usually deploy several firewalls to control access to their internal networks from the Internet and their extranet security threats. Several resources manage firewalls and look at log files generated by firewalls containing a rich set of events crucial for enterprise risk management. Until now, security administrators had to manually sift through multiple firewall log files and look for intrusion data. This approach is laborious, error-prone, and overwhelming to security administrators because of the sheer volume of log entries. The Tivoli Risk Manager Firewall Management feature enables customers to manage their firewall logs centrally. Tivoli Enterprise Console running with Risk Manager correlation server consolidates and monitors events from multiple firewalls. Customers use Tivoli Enterprise Console as the centralized event management server to manage their firewall deployments. Tivoli Risk Manager monitors firewall log files in real time and forwards

them to Tivoli Enterprise Console. Individual firewall alerts are grouped into event classes and archived in a relational database, making it easy for administrators to access them through Crystal Reports or TEDW.

Tivoli Risk Manager implements firewall rules that correlate and aggregate alerts issued by multiple firewalls and presents the alert information in an intuitive, easy-to-use, and manageable fashion. System administrators use the Tivoli enterprise console to manage firewall alerts, close or open alerts, issue trouble tickets, and invoke predefined firewall tasks included with Tivoli Risk Manager. Role-based access control is used to delegate specific firewall administration and management tasks to specific administrators.

For example, events from extranet and partner firewalls will be managed by one group, and Internet firewalls will be managed by a different group of administrators. Tivoli Risk Manager includes a number of tasks to automatically respond to firewall alerts, such as:

- Paging an administrator
- Invoking a firewall action to reset a TCP connection
- Dynamically adding firewall filter rules to block networks or hosts from connecting to the enterprise.

Tivoli Risk Manager supports Check Point FireWall-1 and Cisco PIX Firewall. High-level toolkits are available to integrate other firewalls into Tivoli Risk Manager.

Managing Check Point FireWall-1

Tivoli Risk Manager supports Check Point Firewall-1 and the VPN-1 virtual private network. The firewall adapter for Check

Point monitors the firewall log files on the Firewall-1 management station in real time, formats the log file to events, summarizes the events, and forwards the summarized event. Risk Manager Agent generates an incident through DCE and sends it to the Tivoli Enterprise Console server.

Managing Cisco PIX Firewall

Tivoli Risk Manager supports Cisco PIX Firewall. The firewall adapter for Cisco PIX Firewall monitors the firewall log files on the Cisco management station in real time, formats the log file to events, summarizes the events, and forwards the summarized event to DCE. The Risk Manager Agent forwards the incident to the Tivoli Enterprise Console server.

21.5.2 Intrusion detection

Intrusion detection is an essential prerequisite to enterprise risk management. Customers are deploying several variants of intrusion detection systems:

- Network intrusion detection systems are being deployed to monitor unauthorized access and network-level intrusions.
- Host intrusion detection systems are being deployed to deal with host-level intrusions.
- Application-level intrusion detection is gaining popularity for detecting specific intrusions at the application level that are not detected by the network or host-level components. For example, Web servers are likely to have a real-time intrusion detection engine for HTTP (or HTTPS) application traffic.

Tivoli Risk Manager supports leading network intrusion detection solutions from Interactive Session Support (ISS) (network and host) and Cisco. In addition, Tivoli Risk Manager includes optional network and host-level intrusion detection and real-time application-level intrusion detection for Web servers. Refer to *Risk Manager Release Notes* for supported adapters.

Managing ISS RealSecure Network Engine

Tivoli Risk Manager supports both ISS RealSecure Network Intrusion Detection and the ISS RealSecure Host Intrusion Detection System. The RealSecure adapter monitors the log files generated by RealSecure in real time, and formats the log files to events. Then Risk Manager Client summarizes the event and forwards the summarized event to Agent. Risk Manager Agent generates incident and forwards it to the Tivoli Enterprise Console server using secure Framework communications. Alerts from multiple RealSecure Intrusion Detection Systems are consolidated into an event group within Tivoli Enterprise Console and tagged with the appropriate criticality. System administrators use the Tivoli desktop console to manage diverse intrusion detection alerts with a single console rather than having to learn and use a new console for each intrusion detection system. More important, alerts from RealSecure are used by the Enterprise Intrusion Detection feature that enables system administrators to precisely identify patterns of threats, reduce false alarms, and manage their intrusions with a higher degree of assurance.

Managing Cisco SecureIDS

Tivoli Risk Manager supports the Cisco SecureIDS Network Intrusion Detection System. The Cisco SecureIDS adapter monitors the log files generated by RealSecure in real time

and formats the log files to events. Risk Manager Client summarizes the events and forwards the summary to DCE. Risk Manager Agent generates an incident and forwards it to the Tivoli Enterprise Console. System administrators use the Tivoli desktop console to manage diverse intrusion detection alerts with a single console rather than having to learn and use a new console for each intrusion detection system. More important, the Enterprise Intrusion Detection feature uses alerts from Cisco SecureIDS to help system administrators precisely identify patterns of threats, reduce false alarms, and manage intrusions with a higher degree of assurance using enterprise correlation.

21.5.3 Application intrusion detection management

As more applications become Web-enabled, the probability of attacks and intrusions on these applications increases. Web servers, Java applications, and databases must be secured and monitored to ensure their integrity and availability. Also, attacks increasingly use a secure transport, such as SSL, to render traditional network intrusion detection systems and firewalls incapable of defending against them. This opens up a new class of application-level intrusion detection systems that understand the application-level protocol and monitor threats and intrusions on these applications.

Managing Web server intrusions using Web IDS

In today's e-business environment, protecting the Web infrastructure is one of the most serious challenges facing enterprises. Building a loyal customer base requires the Web infrastructure to be secure, highly available, and reliable, and it must ensure the confidentiality, integrity, and privacy of customer transactions. High-profile denial of

service attacks on Internet portal sites have shown how easily these threats can be carried out to destroy brand equity and cause a loss of customer confidence.

Tivoli Risk Manager includes an intrusion detection system feature called Web IDS. Web IDS is an essential component of enterprise risk management that enables organizations to monitor unauthorized intrusions and various forms of attacks on Web servers. Hackers use sophisticated tools that target URL-level attacks on public Web sites using HTTP or HTTPS (using HTTP over SSL). These types of attacks cannot be blocked by the firewall and frequently bypass monitoring by network intrusion detection tools. An application-level intrusion detection component, such as Web IDS, detects these attacks and enables security administrators to implement incident management and containment.

Web IDS detects and manages attacks on a number of Web servers, including Netscape/iPlanet, Microsoft IIS, Domino, Apache, IBM HTTPS, and Tivoli Access Manager. By integrating with Tivoli Access Manager, organizations allow authorized access for their business constituents and ensure that unauthorized access attempts, attacks, and intrusions are monitored and responded to in real time.

Virus management

One of an IT manager's top concerns is protecting the enterprise from virus threats. Although many organizations have some type of antivirus solution deployed, many are still susceptible to virus attacks, which involve new viruses every day.

Vulnerabilities remain because of a lack of ongoing virus management to ensure that all desktops and systems have the most recent virus software, that virus signature files are

updated, and that desktops not complying with enterprise virus policies are identified and acted on immediately.

Managing antivirus client alerts

Using the Tivoli Desktop feature in Tivoli Risk Manager, system administrators manage Norton AntiVirus and McAfee alerts. The adapters summarize the event and forward it to Risk Manager Agent to generate an incident. The incident is then forwarded to the Tivoli Enterprise Console server.

With Tivoli Risk Manager, security administrators manage Symantec Norton AntiVirus and McAfee alerts from the Tivoli Enterprise Console. Examples of virus alerts include:

- Identify known viruses detected on different desktops.
- Identify unknown viruses detected on different desktops.
- Identify virus definitions that are out of date.
- Identify virus-like activity.

21.5.4 Managing host intrusions using Tivoli Host IDS

Tivoli Host IDS is a host-level intrusion detection system that manages host-level intrusions throughout the enterprise.

Managing alerts from UNIX servers

Frequently, the syslog facility within UNIX will be configured to log useful information about different subsystems within UNIX, such as the UNIX kernel, user processes, TCP/IP subsystem, and devices. The events recorded by syslog

form an important basis for detecting intrusions on host systems. With the syslog facility, Tivoli Risk Manager includes a UNIX adapter that monitors intrusions on UNIX systems. Events are forwarded to the Tivoli Enterprise Console server through Risk Manager Agent and DCE.

Managing Windows servers

The event log facility within Windows is configured to log useful information about different subsystems within Windows, such as the operating system, user processes, remote access, applications, TCP/IP subsystem, and devices. The events recorded by the event log facility form an important basis for detecting intrusions on Windows NT and Windows 2000 and 2003 servers. The event log facility in Tivoli Risk Manager includes a Windows adapter that monitors for intrusions on servers. Events are forwarded to the Tivoli Enterprise Console server through Risk Manager Adapter and DCE. Incidents are correlated with events from other adapters according to Tivoli Risk Manager rules. There are different format files for different Windows versions.

21.5.5 Risk assessment

Correlation for enterprise risk management is the industry's first solution that provides cross-product, cross-platform intrusion management and risk assessment capability. This unique feature of Tivoli Risk Manager provides a correlated view of enterprise intrusions and violations. For example, a business has deployed 50 intrusion detection systems (a combination of network-based, host-based, and application-based engines). Using Tivoli Enterprise Console rules, events from several intrusion detection engines go through a process of duplicate elimination, alert summarization, and distributed correlation to identify patterns of threats. Each pattern of attack is referred to as a situation. By classifying

thousands of alerts into a few precise situations, security administrators quickly gain valuable insight into threats and attack patterns that are monitored by different intrusion detection engines. Native Crystal Report tool reports on Risk Assessment in the enterprise.

Security analysts use the correlation for enterprise risk management feature to:

- Reduce or eliminate false positives by correlating alerts from different sources. Correlation provides a higher degree of assurance in reported alert information and weeds out misleading false alarms.
- Identify attack patterns, single high-profile attacks, and distributed denial of service attacks. Attacks are classified into situations that enable administrators to quickly pinpoint attack patterns.
- Provide a summary view of intrusion data that enables administrators to comprehend the alert data in a meaningful way.

Incidents

An incident is an event representing a set of Risk Manager_SensorEvents with combined rm_Level attributes that have reached the configured threshold within the configured sliding-window time frame. Events received from adapters and clients are correlated by the Risk Manager Event Server. Risk Manager uses a state-based correlation engine to search for patterns of activity. Based on suspicious activity detected, incident events are generated and then forwarded to the Tivoli Enterprise Console server for further processing.

Note This first-level correlation process is deployed in a

distributed fashion by installing one or more Risk Manager Distributed Correlation Servers, which then forward the incident events to the Risk Manager Event Server.

As part of incident identification, the agent also monitors:

- The source of the activity
- The destination (or target) of the activity
- The class category of the activity

What are rm_Level attributes?

The *rm_Level attribute* is a value assigned to each Risk Manager_SensorEvent by the adapter or sensor that generated the event.

What is a sliding window?

In this context, a sliding window is a time period during activity that is monitored by the agent. The start time of the activity is automatically adjusted so that only events received within the configured time period are actively being monitored.

What is a class category?

A class category is an association of Risk Manager_SensorEvent events with a known type of intrusion activity. A class category is assigned to each Risk Manager_SensorEvent based on the type of activity represented by the event. The assignment of class category is typically based on the inheritance hierarchy of the event. Specific event classes are assigned to a category separate from its inheritance hierarchy.

Refer to the *IBM Tivoli Risk Manager Administrator's Guide Version 4.2*, GC32-1323, for further information.

Types of incidents

Incidents that are generated by Risk Manager Agent and then forwarded to DCE for further processing are:

- RM_Cat_Incident
 - Varied activity within a specific class category. This activity originates from more than one source host and targets more than one destination host.
- RM_Dst_Incident
 - Varied activity targeting one destination host. This activity originates from more than one source host and represents more than one class category.
- RM_DstCat_Incident
 - Varied activity targeting one destination host within a specific class category. This activity originates from more than one source host.
- RM_Src_Incident
 - Varied activity from one source host. The activity targets more than one destination host and represents more than one class category.
- RM_SrcCat_Incident
 - Varied activity from one source host within one class category. The activity targets more than one destination host.
- RM_SrcDst_Incident

Varied activity from one source host targeted at one destination host. The activity is within more than one class category.

- RM_SrcDstCat_Incident

Very specific activity from one source host, targeted at one destination host, within a specific class category.

What events contribute to an incident?

Any well-formed event inherited from Risk Manager_SensorEvent with the rm_Correlate=yes attribute might contribute to an incident. Well-formed events have the following attributes set:

- rm_SensorType
- rm_SensorHostname
- rm_SensorIPAddr v rm_SourceHostname, rm_SourceIPAddr, or both
- rm_DestinationHostname, rm_DestinationIPAddr, or both
- rm_Level

These attributes are set by the sensor or adapter or have a valid default value assigned from the BAROC file. Additionally, the rm_Level attribute must be greater than 0.0 for an event to contribute to an incident. Events from trusted hosts do not contribute to incidents.

Second-level correlation: Incident Groups

This is a top-level correlation process that runs on the TEC server. Incident events received from one or more state-based Risk Manager correlation engines are processed further to generate Incident Group events based on sustained suspicious activity patterns. Incident Group events represent an aggregation of multiple incident events.

An incident is generated when a threshold for a defined type of activity is exceeded. When the incident is dispatched, the correlation engine resets the counter to 0. If events keep arriving, a new incident will be generated, after the threshold is exceeded again. This behavior causes multiple incidents to appear for a single attack.

Tivoli Risk Manager uses Incident Groups to reach the objective of displaying only one alert for each attack. If two or more incidents are received by the Event Server, an Incident Group is created. This Incident Group contains the data from all contributing incidents.

As new incidents arrive, the data in the Incident Group is updated, reflecting the accumulated severity and scope of the attack.

Using the Web application, it is possible to drill down from an Incident Group to the associated incidents and Sensor Events.

21.5.6 Tasks for enterprise risk management

Tivoli Risk Manager includes a variety of tasks to quickly resolve security problems, such as:

- Inhibit a connection to or from a specific IP address on the firewall.

- Inhibit and close any existing connections to or from a specific IP address on the firewall.
- Cancel a previously enabled action on the firewall.
- Cancel all previously enabled actions on the firewall.
- Terminate a user process on a server,
- Suspend a user account on a server.
- Scan or delete a virus on a desktop.

21.5.7 The value of enterprise risk management

Finally, before we start going into the technical details with the next chapters, we want to summarize the basic Risk Manager values once again.

Modular deployment

Enterprise risk management offers flexibility for customers to pursue a modular approach to managing intrusions. For example, a customer interested in intrusion detection management can quickly leverage Tivoli Risk Manager to implement enterprise intrusion management as follows:

1. Install and configure a Tivoli Management Region (TMR) on one machine.
2. Install and configure Tivoli Enterprise Console.
3. Install and configure Risk Manager Event Server, Risk Manager Client, Risk Manager Distributed correlation server, and Risk Manager Gateway. These components can be installed in any order.

4. Install & configure optional components such as reporting components.
5. Install Endpoint on the system holding Risk Manager Adapter, but only if TME Adapter is installed.
6. Install and configure Tivoli Risk Manager adapters for the network intrusion systems that need to be managed.

These steps have an estimated completion of three days, assuming five network intrusion systems need to be managed.

With the event management infrastructure in place, the power of an integrated risk management framework is now been demonstrated.

The steps required to integrate and manage additional servers, such as UNIX (50 UNIX servers) and Windows NT/2000 (100 Windows NT servers), are:

1. Distribute the Tivoli Management Agent and the Tivoli Host IDS (using the adapter profile) on the UNIX and Windows NT/2000 servers from the Tivoli desktop console.
2. Create and distribute the appropriate host security configuration on all of the managed servers from the Tivoli desktop console. Configuration settings control the intrusions detected and alerted by the Tivoli Host IDS system.

This illustrates the power of a scalable Framework across the enterprise. Customers start with a small, manageable environment, such as managing intrusion detection systems or managing firewalls with a single TMR. With the risk management infrastructure in place, the solution then can

be expanded to support managed technologies (using appropriate profile adapters), such as routers, Web servers, UNIX servers, and desktops.

Support for open standards

Tivoli Risk Manager actively promotes, supports, and contributes to the emerging open systems standards in the area of intrusion detection, including the following:

- The Common Vulnerabilities and Exposures (CVE)^[1]
- The Intrusion Detection Exchange Format (IDEF)^[2]

The CVE list is a dictionary of standardized names for vulnerability and other information. CVE standardizes the names for all publicly known vulnerabilities and security exposures.

IDEF is a common intrusion data model specification that describes data formats that enable communication among intrusion detection systems and communication between intrusion detection systems and management systems.

Tivoli Risk Manager uses an IDEF-compliant protocol for communication with ID sensors. The IDEF standard integrates new security endpoints from multiple vendors into an enterprise risk management platform, such as Tivoli Risk Manager. An IDEF draft is available for review from the Intrusion Detection Working Group (IDWG) in the IETF.

Tivoli Risk Manager toolkit

Tivoli Risk Manager is built on top of Tivoli Enterprise Console, and it leverages the Tivoli Enterprise Console API for integration.

To integrate with Tivoli Risk Manager, an alert generator:

1. Sends IDEF-compliant events to the Tivoli server via an event generator
2. Provides response units to enable the administrator to react to attacks or resolve exposures
3. Provides a Tivoli-compliant mechanism to install and configure the event generators and response units

[1]<http://www.cve.mitre.org>

[2]<http://www.ietf.org/html.charters/idwg-charter.html>

21.6 Conclusion

The knowledge economy is causing fundamental changes in enterprises as they transform themselves to extend their traditional business models to the Web. Web companies are creating new ways of doing business on the Internet. All of these changes are exciting but fraught with challenges, and companies must understand the challenge of being on the Internet. The open Internet environment combined with lack of security makes companies unwitting targets of intrusions, such as virus threats, denial of service attacks, and unauthorized access. These threats are real. In the Computer Security Institute's 2003 CSI/FBI Computer Crime and Security Survey, theft of proprietary information caused the greatest financial loss (US\$70,195,900 was lost, with the average reported loss being approximately \$2.7 million). The second most expensive computer crime among survey respondents was denial of service, with a cost of \$65,643,300. Virus incidents (82%) and insider abuse of network access (80%) were the most cited forms of attack. [Figure 21-7](#) shows the number of IT security attacks over the past few years that were reported during the survey.

How Many Incidents?						
By percentage (%)	1 to 5	6 to 10	11 to 30	31 to 60	Over 60	Don't Know
2003	38	20	more:16	0	0	26
2002	42	20	8	2	5	23
2001	33	24	5	1	5	31
2000	33	23	5	2	6	31
1999	34	22	7	2	5	29

2003: 316 Respondents/63%, 2002: 312 Respondents/64%, 2001: 348 Respondents/65%, 2000: 392 Respondents/64%, 1999: 327 Respondents/63%

How Many From the Outside?						
By percentage (%)	1 to 5	6 to 10	11 to 30	31 to 60	Over 60	Don't Know
2003	46	10	13	0	0	31
2002	49	14	5	0	4	27
2001	41	14	3	1	3	39
2000	39	11	2	2	4	42
1999	43	8	5	1	3	39

2003: 316 Respondents/63%, 2002: 312 Respondents/64%, 2001: 348 Respondents/65%, 2000: 343 Respondents/53%, 1999: 280 Respondents/54%

How Many From the Inside?						
By percentage (%)	1 to 5	6 to 10	11 to 30	31 to 60	Over 60	Don't Know
2003*	45	11	12	0	0	33
2002	42	13	6	2	1	35
2001	40	12	3	0	4	41
2000	38	16	5	1	3	37
1999	37	16	9	1	2	35

2003: 318 Respondents/62%, 2002: 289 Respondents/57%, 2001: 348 Respondents/65%, 2000: 392 Respondents/64%, 1999: 327 Respondents/63%

Figure 21-7: How many incidents? How many from outside? How many from inside?

Figure 21-8 shows the major point of Intrusion attacks reported in a survey conducted by the Computer Security Institute. The solution proposed in the next chapter addresses the major points of intrusion that were identified.

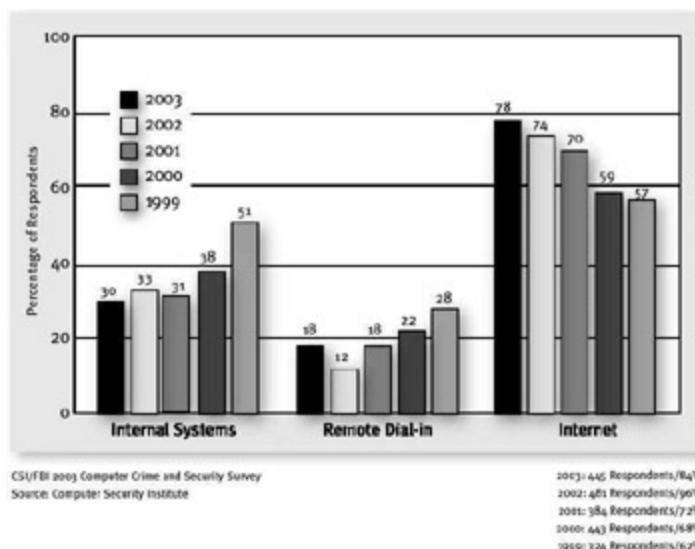


Figure 21-8: Internet connection increasingly cited as a frequent point of attack

Figure 21-9 also shows the likely sources of attack from the CSI 2003 survey.

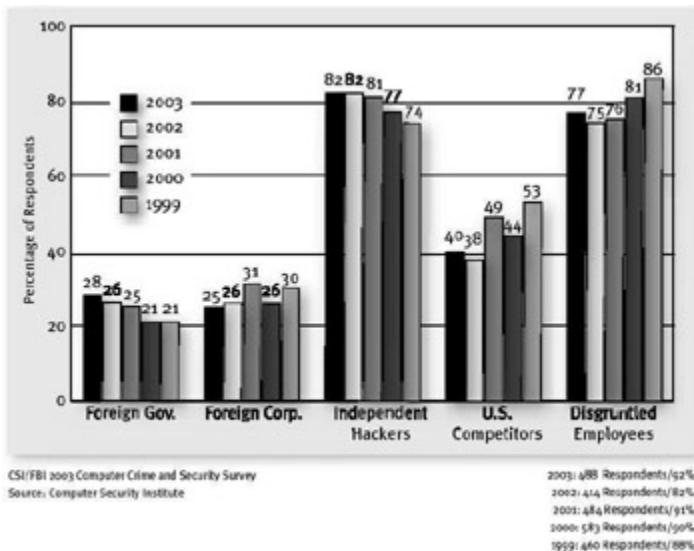


Figure 21-9: Likely sources of attack

Integrated management of security alerts, practices, and policies is imperative to mitigating enterprise risks. An enterprise risk management solution, such as Tivoli Risk Manager, enables security analysts to centrally manage their enterprise security and use intelligent correlation and decision support to quickly identify users, networks, and hot spots (such as critical systems) and fix vulnerabilities. The business implications are significant if the company's e-mail servers or Web servers go down or are compromised. Having a single point of control to deploy an enterprise risk management solution for security across the enterprise helps mitigate risk, ensures business continuity, and is a compelling return on investment strategy that companies will not ignore.

Chapter 22: Building a Centralized Security Audit Subsystem

As described in [Chapter 2, “Method for Architecting Secure Solutions”](#) on [page 15](#), the purpose of the security audit subsystem 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 audit 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. Tivoli Risk Manager addresses both of these requirements. The Event Console and rules engine alert security managers to problems by correlating thousands of events into more specific incidents to identify attacks. Crystal Reports help in near-term reporting, and the Tivoli Enterprise Data Warehouse enablement pack of Risk Manager allows for trend analysis of security events, enabling the security manager to understand vulnerabilities and initiate preventive actions.

We look at the IT setup in order to describe the audit solution approach. The scenario depicts our already introduced small medium business (SMB) enterprise, Stocks-4u.com, that is a wholly owned subsidiary of a major brokerage company, Medvin, Lasser & Jenkins (ML&J). 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.

22.1 Scenario profile

Stocks-4U.com is an upcoming online trading company in the early stages of implementing an e-business infrastructure facing the Internet. This scenario goes back to the early days of Stocks-4u.com. The current information is distributed between two Web servers. Functions of the individual Web servers are:

- **Web server one**

The first Web server is designated for customers and interested parties. It is primarily for information retrieval, with organization profile, product and offering information, and technical documents available for download.

- **Web server two**

The second Web server is designated for partners. It provides more detailed information about the products and enables partners to post forms and documents to the company. Partners update their requirements, which become confidential data within the organization.

Because both Web servers technically belong to different application project teams in the company and have been developed concurrently, they are based on different platforms: Netscape Enterprise Web Server and IBM HTTP server. 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 intranet in one physical location have been set up. The employees are running Windows operating systems on their desktops. For a general overview of the initial network layout, see [Figure 22-1](#) on [page 561](#). The network operators have the responsibility for the firewalls and antivirus software. The security administration is handled by the application project teams.

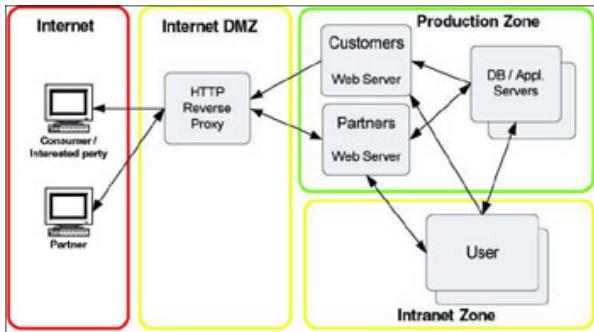


Figure 22-1: Initial IT architecture for Stocks-4U.com.

22.1.1 Security-related problem

This company's infrastructure is very typical for small companies just starting with some Internet exposure. They have a small staff and limited technical expertise. In their current configuration, the company is unsure how to validate the security of its environment. They are aware of more tools in the marketplace, such as PKI and Intrusion Detection, but have no skills in-house to attempt installation, and no overall security plan to convince their senior management to obtain the required resources.

22.1.2 Business requirements

At this stage, the CEO is looking for a way to validate the security of the network, and thus enhance customer and partner confidence. Having invested in a variety of security products (firewalls, IDS, and routers), there is still a lack of comfort that the environment is secure. After a recent virus attack, antivirus software was put in place, but his IT staff cannot assure him that people are using it. 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. The CEO is concerned about the amount of investment required with no apparent way to measure the effectiveness of the solution.

22.1.3 Business design

To meet the needs of the business, the audit flow structure depicted in [Figure 22-2](#) must look at the audit events from the security tools, be able to identify real threats and attacks, and define actions to be taken

automatically or by manual intervention. Identifying real threats from the volumes of alerts generated by multiple security sensors will make the environment more secure and more manageable. Predefined actions 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.

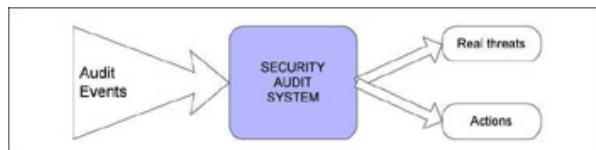


Figure 22-2: Audit flow structure

22.1.4 Security design objectives

The primary objective of the security audit subsystem is to enhance the security management function through the collection, analysis, and archiving of security data generated by the security environment in both real-time and historical modes. The audit subsystem must be able to isolate real security alarms from the vast flow of security events and correlate events from several sources. The audit subsystem 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. 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 more effective by delegating common tasks to operator-level personnel or even automating responses to common situations.

22.2 Security audit subsystem

A security audit subsystem is responsible for capturing, analyzing, reporting, archiving, and retrieving records of events and conditions.

[Figure 22-3](#) shows a use case model of a security audit function. 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 security audit function. The logical view breaks down the audit process into distinct functions.

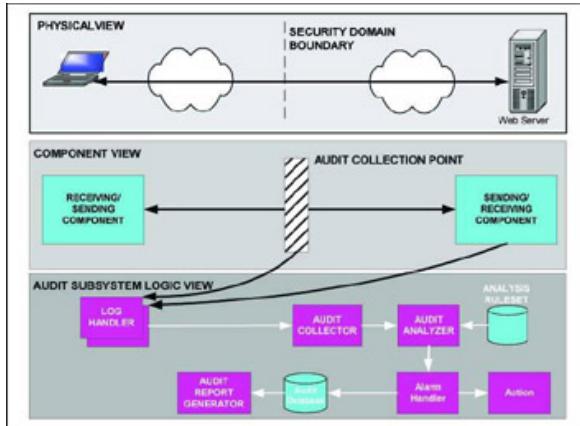


Figure 22-3: Audit subsystem

The *log handler* is usually a standard part of a component, 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.

The *audit collector* converts or reformats the events from the generic log handler into a format usable by the audit subsystem. It also summarizes the events depending on the rules so that the number of events flowing from audit collector to analysis engine becomes streamlined and network traffic due to the flood of events is minimized.

The *audit analyzer* (or analysis engine) receives events from the audit collector. It uses security rules based on an artificial intelligence engine and filters to correlate the events. This is the core function of the audit subsystem, and its effectiveness depends on the *analysis ruleset*.

The output of the analysis engine is sent to the *alarm handler* to perform two functions. First, to generate an alarm or initiate an *action* (if necessary) and route it to the correct application or device, and second, to store all records in a centralized *audit database* that will be used for audit reports generation.

An *audit report generator* executes extensive analysis on long-term data stored in the audit database. These reports help understand general vulnerabilities within the environment that do not generate incidents.

Note The analysis 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.

22.2.1 Audit subsystem at Stocks-4U.com

To apply this subsystem within their IT infrastructure, Stocks-4U.com would first need to identify which components in their configuration are generating security-relevant events. Next, an audit handler has to be added to each of those components to collect events and forward them to an analyzer engine. In [Figure 22-4](#) on [page 565](#), we show the audit handlers in the customer's configuration. As shown, the components to be audited are:

- Cisco router: To collect configuration change data, connection information, and exception or error events.
- Firewalls: The audit handler should enable you to collect information about flows as well as accepted or denied connections among parts of the network.
- Web servers: To summarize the activity of the Web server and to track down events such as unsuccessful logons, configuration changes, or long URL attacks.
- Servers: To collect access control exceptions from the operating systems and applications.
- Users: To gather data on viruses detected by antivirus software.

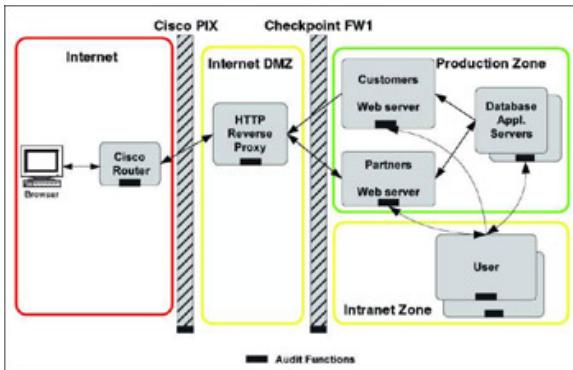


Figure 22-4: Audit functions

22.2.2 Risk Manager in the audit subsystem

[Figure 22-5 on page 566](#) depicts how the Risk Manager components are mapped to the logical functions on the security audit subsystem we discussed using the points of audit functions marked in [Figure 22-4](#) to collect the audit information.

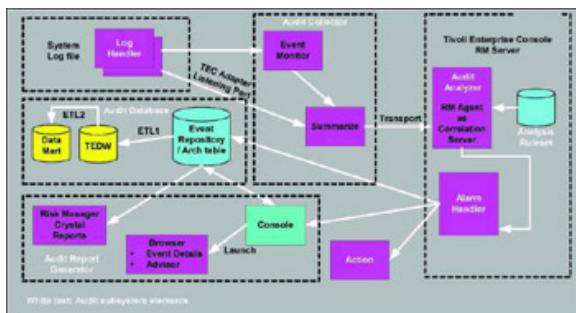


Figure 22-5: Logic view with Risk Manager components

Log handler

The log handler is the component that determines which events in a function should be reported. In a firewall, for example, the log handler could identify an unauthorized access attempt as an event to be reported and dispatch it. Other functions simply store all events in a log, such as the system log for an operating system. For these environments, Risk Manager provides *sensors* to read the logs and extract events for the audit subsystem. The sensors provided are for Web-based intrusion detection, host-based intrusion detection, and network-based intrusion detection. Each sensor uses the respective logs to detect attacks and suspicious activity. There are two ways to forward an event from the log handler to audit collector:

1. Use a TEC adapter and forward the events to the audit collector using an *events listening port* for summarization.
2. Forward the event, such as an SNMP trap, to an event monitor running on the audit collector, which then forwards the event for summarization.

Audit collector

The audit collector function is performed by Risk Manager *adapters*. The adapter takes the logs from the system and formats them in the correct way to be sent to the Risk Manager server DCE. An important strength of the Risk Manager is the Event Integration Facility (EIF), a toolkit for building adapters using an application programming interface. In addition, existing adapters can be configured to invoke a summarization function that reduces high-volume floods of duplicate events into a relatively low-volume set of summarized events. This summarization reduces the volume of events sent to the Risk Manager server, greatly reducing network traffic.

Note It is advisable to place the adapter very near to or on the same system where events occur in the first place if possible, so as to reduce the events flow on the network.

Audit analyzer

The Risk Manager adapters send events to the Tivoli Enterprise Console (TEC) event server running the Distributed Correlation Engine. The Risk Manager agent, running as a correlation server on the TEC server, analyzes events and creates *incidents* based on the class and destinations (target and source) of the events. Thousands of events can trigger a single incident based on the rules defined. An incident is the result of applying the Risk Manager rules to correlate events received from the different event sources throughout the enterprise. Incidents identify patterns of threats that provide invaluable insight into how the enterprise is being targeted. Based on the incident and TEC rules, an *incident group* is created with further correlation.

Analysis ruleset

The Risk Manager rules and event format definitions are added to the TEC rules and definitions. These rule files are the key to recognizing attack patterns, and they are the strength of Risk Manager.

The rule files are as follows (RMINSTDIR represents the installation directory for Risk Manager on your machine):

- RMINSTDIR/etc/tec/rules/riskmanager.rls
- RMINSTDIR/etc/tec/rules/boot.rls
- RMINSTDIR/etc/tec/rules/hostname_remap.rls

- Configuring the riskmanager.rls file:

The riskmanager.rls file contains rules to route RM_SensorEvent events that have not been included in correlation to the local agent, and perform incident group processing.

- Configuring the boot.rls file:

The boot.rls file contains rules to activate the configuration options specified in the riskmanager_config.pro file, ensure that the Tivoli Risk Manager rules and configuration settings are active. If not, an RM_InputError event is generated at the Tivoli Enterprise Console server.

- Configuring the hostname_remap.rls file:

The hostname_remap.rls file is a sample rules file containing rules to set the hostname attribute of fully processed RM_SensorEvent events to the rm_SensorHostname attribute value, and set the adapter_host attribute to the composite string that Tivoli Risk Manager sets as the hostname attribute.

Tivoli Risk Manager sets a composite string as the hostname attribute to enable the attack information to be viewed on the event console. This composite string contains:

- The attack category
- The source of the attack
- The destination of the attack

Alarm handler

The event server sends incidents and events to the TEC console for display. The TEC event server, in addition to correlating events and generating incident groups, initiates actions. An action can be an automated task in reaction to certain events or incidents. The task runs a local script that starts some actions on the right hosts.

Audit database

The Risk Manager DCE and event server stores events, incidents, and incident groups within a relational database.

Audit databases for Risk Manager consists of these tables:

- Event Repository: This consists of all events reaching to TEC from all of the integrated modules.
- The Risk Manager agent DCE stores summarized events causing incidents into an archive table created within a relational database. The Risk Manager event server stores incident group events and individual events in an event repository. Risk Manager supports many third-party databases, such as IBM DB2, Oracle, Sybase, and Microsoft SQL.
- TEDW: The Tivoli Data Warehouse module has a separate database for Risk Manager reporting.
- Data Mart: This is a repository maintained by the TEDW server to store the formatted events from the TEDW database for presenting the reports. This data is used by the Crystal Reports provided with Risk Manager.

Console

The Risk Manager console filters which incidents and events are displayed by the priority of the events (all, warnings, critical, and fatal). The console of the audit subsystem is provided by TEC. You can arrange the views by groups of events, by operator groups, filtering the view of the events, and so on. The console is a browser-based Web application that presents event details for individual incidents and incident groups. It also offers an advisor function to provide more detailed system information if a Tivoli Inventory Module is implemented on the monitored systems as well.

Audit report generator

The report handler is a collection of these components:

- TEDW Reports: Pre-defined reports are available for Risk Manager from the TEDW enablement pack for Risk Manager. These reports are historical trend analysis reports. TEDW reports are accessible by using a standard Web browser interface.
- Crystal Reports: Risk Manager 4.2 provides an out-of-the-box, prepared set of Crystal Reports. These reports present the data from the event repository archived table. Crystal Reports provides predefined near-term reports. Crystal Reports are also available through a Web browser interface.

22.2.3 Integration of Risk 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 audit subsystem. Multiple security management consoles spread across multiple zones in the environment do not support a truly secure setup. To implement risk management at Stocks-4u.com, management functions should be grouped into a single zone. In [Figure 22-6 on page 570](#), all management functions, including the Risk Manager event server, are placed within one zone. The exception is the Risk Manager Gateway with SNMP Adapter in the Internet DMZ, which allows SSL communication through the firewall instead of opening additional ports on the Checkpoint firewall for SNMP traps from the Cisco router to pass through.

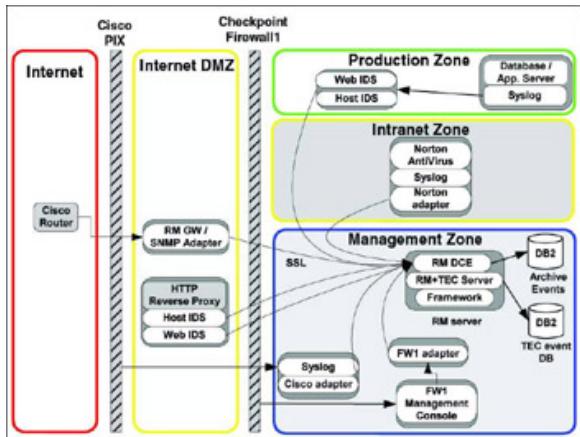


Figure 22-6: Risk Manager flow

Environment

The TEC server requires the Tivoli Management Framework as the underlying infrastructure. Stocks-4U.com is currently running an environment without the Tivoli Management Environment® (TME). In order to use Risk Manager, Stocks-4U.com has to install the Framework on the Risk Manager server, which becomes the Tivoli Management Region server (TMR). All Risk Manager adapters are installed in a non-TME mode. In this case, there is no need to set up the Tivoli environment all over the network (endpoints, managed nodes, gateways, and so on). The TEC and Risk Manager agent both need a relational database to store the events sent by the distributed components and the sensor events causing incidents. Risk Manager does not come with a database. An IBM DB2 database could be installed on the TMR server. In order to increase the overall performance, you should install the database on a dedicated host, install a Tivoli Managed Node on that system, and set up the RIM and the database connectivity between the TMF/TEC/Risk Manager server and the database server. A separate table for the archived events can be created in the same or a separate database. Risk manager DCE uses JDBC to communicate with the archive tables using a local RDBMS client to store sensor events causing incidents.

Communications with the Risk Manager server

In this section we discuss the different ways Risk Manager and the other products exchange audit data.

CISCO router

The Cisco router generates a Simple Network Management Protocol (SNMP) trap when the configuration changes or when there are unsuccessful logons. The SNMP event is sent directly to a Windows system, either an endpoint in a TME environment or a Tivoli SNMP adapter running on a non-TME machine. The Risk Manager adapter for Cisco routers consists of files that configure the Tivoli SNMP adapter to capture and forward the Cisco router events to the event server for correlation. We install a Risk Manager gateway on the same system so that we do not need to open the SNMP port 162 on the Checkpoint firewall. The Risk Manager gateway communicates using SSL to the Risk Manager Server running on TEC Server for correlation.

Attention This feature should open port 162 on the Cisco PIX firewalls only for the SNMP trap to flow to the adapter system, but not on the Check Point Firewall.

CISCO PIX

The adapter for Cisco Secure PIX Firewall resides on the host to which the Cisco Secure PIX Firewall sensor has been configured to send log messages. This host can be a UNIX or a Windows system. The Cisco PIX Firewall Syslog Server (PFSS) is required for logging to a Windows system host.

Check Point FireWall-1

The Check Point FireWall-1 sends its logs to a dedicated management console. (Data transfer is encrypted.) The Risk Manager adapter uses the OPSEC interface to retrieve the log file stored on the FireWall-1 management console.

Web Intrusion Detection System (Web IDS)

The default configuration for Web IDS uses the Risk Manager EIF in order to send the Web IDS events to the Risk Manager server. You must customize the Risk Manager EIF to use the Web IDS format file in order to perform the proper mapping of Web IDS events into Risk Manager TEC events.

Host Intrusion Detection System (Host IDS)

The Risk Manager adapter for Host IDS maps events that are detected and logged by the Windows or UNIX system logs, into TEC events. The Risk Manager adapter for Host IDS uses the Tivoli log file adapter (syslogd) for UNIX systems or the Tivoli NT Event Log adapter for Windows systems to send events to the Risk Manager server.

Norton AntiVirus

Norton AntiVirus writes events in the Windows system event log. The Tivoli Event Log adapter recognizes the virus-related events sent by Norton AntiVirus on a Windows system and maps them into Risk Manager events, which are then sent to the Risk Manager Agent DCE and then to Risk Manager event server for further correlation.

22.3 Expanding security monitoring

To enhance the security of Web environments, other security tools should be installed. For access control functions, an HTTP reverse proxy server solution such as Tivoli Access Manager is recommended. In order to monitor network traffic (users, customers, and partners), a network intrusion detection system such as Interactive Session Support RealSecure 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 Risk Manager security audit subsystem.

HTTP reverse proxy server

An HTTP reverse proxy server provides single-point management of authentication and access control. Risk Manager comes with an adapter for Tivoli Access Manager's reverse proxy server, WebSEAL, in order to send alerts to the central Risk Manager server.

Intrusion detection

An Intrusion Detection System (IDS) is a scanning system designed for computers (Host IDS), Web servers (Web IDS), and networks (Network IDS). An IDS gathers and analyzes information from various areas 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 IDS 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
- Analyzing system configurations and vulnerabilities
- Assessing system and file integrity
- Recognizing patterns typical for attacks
- Analysis of abnormal activity patterns

Filter function

When the number of security products used within an enterprise increases, the number of single security events also increases. In this case, the audit subsystem must be able to filter events locally before they are sent to the Risk Manager server DCE in order to improve overall throughput.

For this reason, Risk Manager provides a summarization engine. As shown in [Figure 22-7](#), this feature enables you to filter events locally before they are sent to the Risk Manager server DCE. The Risk Manager observer filter function is based on the Risk Manager EIF component and is installed with the adapter or sensor. The Risk Manager observer summarizes events locally and sends only one event to Risk Manager server DCE with the repeat count, so that there is less network traffic and reduced load on the Risk Manager server DCE when it comes to a fast correlation process.

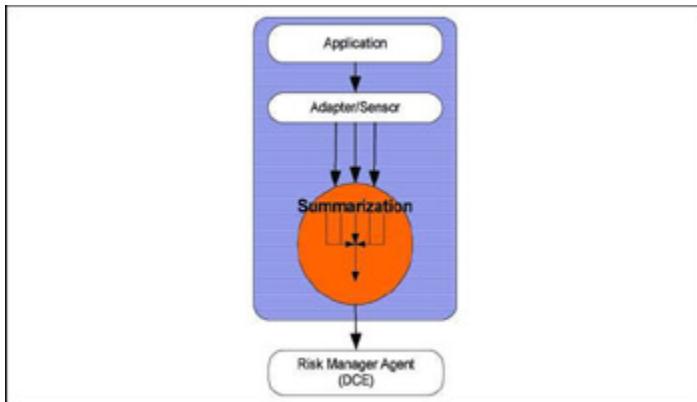


Figure 22-7: Local summarization engine

For more details about the RMEIF function, refer to the *Tivoli Event Integration Facility Reference Version 3.9*, SC32-1241.

22.3.1 Supported tools

Tivoli Risk Manager adapters are now available to customers through the Tivoli Risk Manager Support Web site and are no longer included on the product CD.

This enables new and improved adapters to be distributed independently from new releases of Tivoli Risk Manager and allows customers to download only the adapters they require.

For Tivoli Risk Manager adapters, the latest product updates including sensor signatures, and service information about Tivoli Risk Manager, visit:

<http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliRiskManager.html>

For information about the IBM Tivoli Risk Manager product, visit:

<http://www.ibm.com/software/tivoli/products/risk-mgr/>

For information about other Tivoli systems management products, visit:

<http://www.ibm.com/software/sysmgmt/>

Risk Manager supports these kinds of applications:

- Firewalls
- Web servers
- Intrusion detection systems
- Antivirus products
- Routers
- Operating systems log file events

Risk Manager EIF enables you to develop your own security applications and integrates them into the Risk Manager environment.

22.4 Mapping the solution to the organization

The ability to delegate the audit functions within Risk Manager enables an organization to distribute responsibilities to different administrative people. Security administrators will have the responsibility for customizing the rules files and defining things such as thresholds and categories, while IT operators only see basic security alerts and events. This functional delegation model should be applied according to the individual internal organization of the company.

The purpose of this discussion is to describe the functional responsibilities.

Figure 22-8 depicts the 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 the Risk Manager files 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.

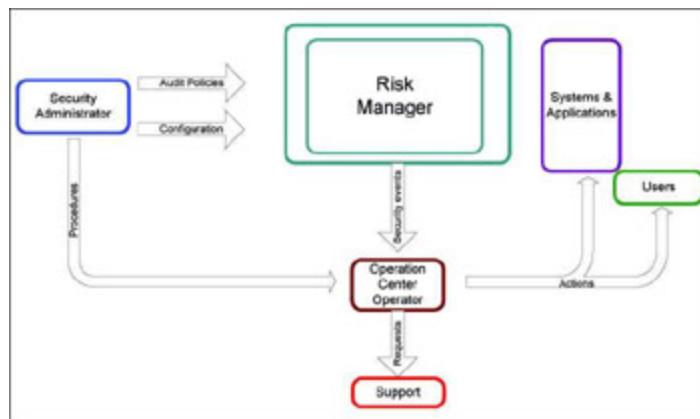


Figure 22-8: Organization flows

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.

22.5 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.

Tivoli Risk 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, Risk Manager is able to 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, Risk Manager also has a powerful Crystal reporting tool for near-term reporting and set of TEDW reports for trend analysis to facilitate preventive measures and planning.

Chapter 23: Extending the Centralized Security Audit Subsystem

This chapter demonstrates expanding the security audit subsystem into a more complex environment. In addition to the real-time security management processes using near-term reporting, historical view of data collected by the audit subsystem, with tasks of data mining and analysis, is explored in this chapter. These views enable IT management to understand the effectiveness of the security system in order to identify areas of potential problems and to take action for improved security. Web-based reports are available over the network.

23.1 Company profile

Stocks-4U.com has been acquired by a large enterprise. After the merger, Stocks-4U.com still maintains their own network, Web site, and applications, including e-commerce, but all connections external to their network flow through the new corporate center. The challenge is how to integrate these entities into a secure corporate environment, with consistent security policies and management processes.

For this discussion, we focus on three locations, where the headquarters is responsible for the management of the enterprise, and each individual location maintains its own network (Web servers, application servers, databases and so on), as shown in [Figure 23-1](#) on [page 579](#). In each location, there is a local administrator available to supervise all of the elements of the local environment, systems, and network, as well as security. Several types of firewalls (PIX, FW1, and IBM) and ISS RealSecure network scanners are already installed, as covered in previous chapters.

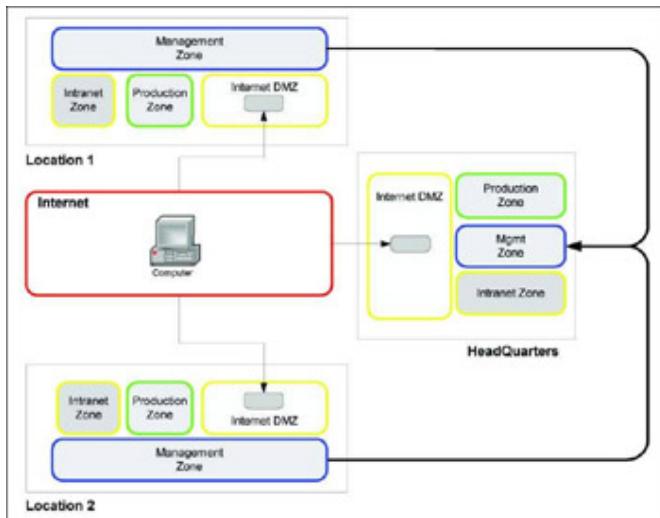


Figure 23-1: Network integration architecture

23.1.1 Business requirements

From the overall enterprise perspective, the goal is to have a consistent level of security across the business. The CEO wants to be assured that there will be no new exposures because they acquired some companies in order to expand the global reach of the enterprise.

As mentioned earlier, security is only as strong as the weakest link, and in this distributed environment, it is desirable to have local operators who act quickly on threats.

At the headquarters location, the administrators need to see the security events from all locations, as well as local events, to be able to recognize attacks across boundaries. There is an obvious concern about the number of events being sent to the headquarters site, so there should be a way to filter and reduce the number of incoming events to the consolidated administrators.

Beyond these day-to-day requirements, the CIO also wants to be assured that the environment is secure and that the proper actions are being taken to react to the threats. He wants to be able to look at each organizational level to see whether any local problems exist.

23.1.2 Business design

In each location, the complete set of audit functions is implemented for routine security monitoring. Because of the new requirements, this scenario introduces the concept of *data mining*, shown in [Figure 23-2](#). This function helps the enterprise to understand the effectiveness of high level decisions and policies in greater detail by summarizing and analyzing the data in many different ways, and allowing for drilldowns to identify individual component problems.



Figure 23-2: Enhanced audit flow structure

23.1.3 Security design objectives

The reduction of the number of events sent to the central management servers is very important in order to forward only enterprise-critical events. Otherwise, the network can be overloaded with audit flow alone, which can be very heavy in an ongoing attack (thousands of events per minute). The security audit product should provide a way to summarize similar events coming from the same source without attenuating the meaning of those events. In addition, local correlation could provide even more meaningful events to be forwarded to the headquarters.

A data mining product is an essential tool in the security management system. Administrators must have the ability to build historical reports and to summarize suspicious activities (per day, per network, per location, and so on). A data mining tool provides the capability to be proactive and to have benchmarks to improve the global network security of the company. This tool helps security administrators to prove the effectiveness of the countermeasures that are taken.

Security administrators decide how to set up the security audit system and which actions are to be taken when a threat is discovered. The audit needs may not be the same for each location, so the security audit product should provide the capability to distribute a general audit configuration to all sites and the flexibility to allow for independent customization of thresholds and actions.

23.1.4 Security audit subsystem

In this expanded enterprise view, all components of the security audit subsystem (log handler, audit handler, analysis engine,

security rules, alarm handler, storage, and console) are implemented within the headquarters, and other locations of the company might have log handlers, audit handlers, and storage for sensor events. This helps in building a centralized management system and keeping a tight control on the event flow from other locations to the headquarters. Refer to 22.2, “Security audit subsystem” on [page 563](#) and 21.4, “Risk Manager architecture” on [page 538](#) for more details about the components.

For the enterprise view, incidents are sent to the headquarters' security management zone for further correlation with events from other locations. Data mining and analysis of raw data (all incident events from all locations) is performed at the headquarters to monitor global security management of the enterprise.

23.2 Risk Manager

The design of Tivoli Risk Manager 4.2 is very scalable, making it easy to implement Risk Manager in a larger enterprise and distributed environment. This section explains the layout for implementing the audit subsystem using Risk Manager in a complex environment.

23.2.1 The distributed environment

With multiple locations and an expanding set of security sensors, it is not advisable to send all events to one Risk Manager server. Any one of the following situations could pose problems:

- The workload of the Risk Manager server or TEC server will be so great that it cannot handle all of the events in a reasonable time.
- The links between locations have a limited bandwidth and are already partially loaded with production non-security data.
- Each location has its own particularity and administrators.

Risk Manager can be configured into a distributed environment that fits the structure of the enterprise as follows:

- One Risk Manager agent with state-based correlation engine per location
- A global Risk Manager server that correlates all local events and incidents as well as optional sensor and incident events received from the locations

This architectural approach, as shown in [Figure 23-3](#), addresses the workload concerns. Only incidents and optional sensor events are sent to the central Risk Manager server, instead of all events. The architecture also provides near-term reporting using Crystal Reports at every location.

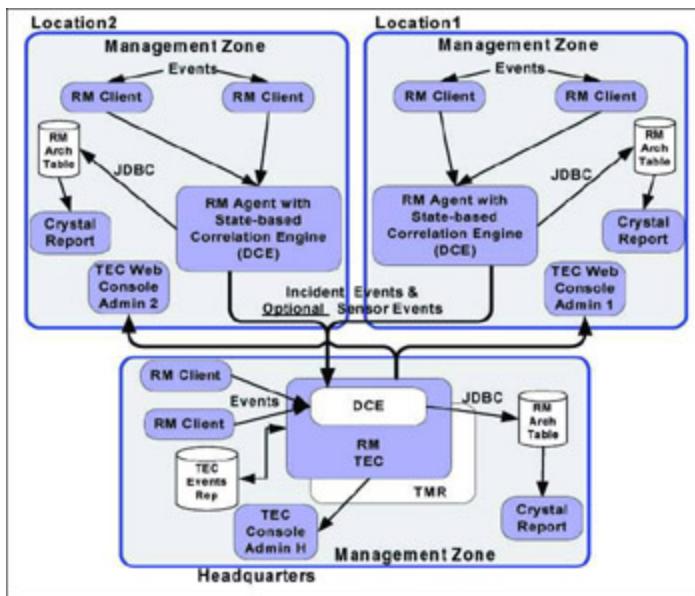


Figure 23-3: Risk Manager and Tivoli Enterprise Console architecture

The Event Integration Facility (EIF) summarization capability reduces the load of the local Risk Manager agent in each location by only forwarding summarized events from the individual sensors and adapters running on Risk Manager clients. All location-specific events are then correlated by the local Risk Manager agent that is running DCE and stored locally within each location's archive database. After correlation, only incidents are forwarded to the headquarters Risk Manager server DCE. The headquarters server correlates these incidents from the remote locations to recognize a distributed attack against the global enterprise. This is a good example of a case where individual policies can be the same but thresholds for

correlation are different for headquarters and locations. Because the global Risk Manager server only receives incidents, the thresholds would be much lower than on the remote servers that receive a flood of events.

This architecture gives us the ability to manage local security events and provides a high level correlation for the headquarters administrator.

Because all events are stored in a relational database, standard SQL queries could be used to extract reports for Risk Manager events. The data mining capability, however, builds the event information into three dimensional views, and provides analysis and reporting features beyond simple SQL. This is achieved with the Tivoli Data Warehouse application.

23.3 Tivoli Data Warehouse

For architecture considerations, there are four components of the Tivoli Data Warehouse:

- Control server
- Central data warehouse
- Data mart
- Report interface

The *control server* is the system that contains the control database for Tivoli Data Warehouse and from which you manage your data warehouse. The control server must be located on a Windows 2000 or Windows NT machine.

The *central data warehouse* server contains only the DB2 databases. In this configuration, no pieces of the Tivoli Data Warehouse software or DB2 warehouse components are needed on this server. Supported operating systems are Windows NT, Windows 2000, AIX, and Solaris.

The same applies to the *data mart server*. For this reason, in a typical configuration, the central data warehouse and the data marts are located on the same database server.

The *report interface server* (or *report server*) provides tools and a graphical user interface to create and display reports that help you analyze the data in your warehouse to answer questions that are important to your business. The report interface utilizes Tivoli Presentation Services.

The report interface requires a DB2 run-time client to access data in the database instances on the central data warehouse, data mart, and control servers. Supported operating systems for the report server are Windows NT,

Windows 2000, AIX, Solaris, and Linux. It performs best on Windows NT and Windows 2000 systems.

Tivoli Data Warehouse aids analysis and decision making in an organization. In addition to Risk Manager, Tivoli Data Warehouse enablement packs are available for Network Management, Configuration Management, Event Management, and more.

Refer to the IBM Tivoli Enterprise Data Warehouse product documentation for more details.

23.4 Tivoli Data Warehouse and Risk Manager

The Tivoli Data Warehouse enablement pack for Enterprise Risk Management component of Risk Manager provides aggregated, historical information about security events reported to the Risk Manager event console. Tivoli Data Warehouse provides the infrastructure for the following:

- Extract, transform, and load (ETL) processes through the IBM DB2 Data Warehouse Center tool
- Schema generation of the central data warehouse
- Historical reporting

The IBM Tivoli Risk Manager warehouse enablement pack aggregates event data from the Tivoli Risk Manager events archive table. The warehouse pack aggregates events based on the same three key attributes (events table column names in parentheses) used in Tivoli Risk Manager correlation:

- Event category (CLASS_CAT)
- Event source token (SRC_TOKEN), hostname (SRC_HOSTNAME), or IP address (SRC_IPADDR)
- Event target hostname (DST_HOSTNAME) or IP address (DST_IPADDR)

To assist in understanding the Tivoli Data Warehouse functions, [Figure 23-4](#) on [page 585](#) depicts basic components within Tivoli Data Warehouse, such as data collection, data manipulation, and data reporting.

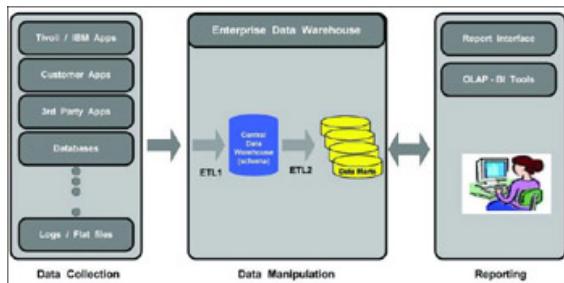


Figure 23-4: Tivoli Data Warehouse basic components

Data collection

The first phase of a warehouse project consists of defining all involved data sources and determining the truly relevant data in our business.

Generally IT departments are responsible for the collection of metrics concerning availability, performances, and utilization of IT resources (systems, applications, and networks). Other departments could provide additional data, such as financial statistics or inventories.

Skills required to select and actually perform data collection vary according the type of data. Generally, a system administrator or an application administrator is directly responsible for the implementation of IT resources monitors or may provide the information required to implement them.

Other than Tivoli Risk Manager, Tivoli software products offer a wide range of solutions for collecting data about IT infrastructure, such as:

- IBM Tivoli Monitoring
- IBM Tivoli Netview
- IBM Tivoli Switch Analyzer
- IBM Tivoli Enterprise Console
- IBM Tivoli Business Systems Manager
- IBM Tivoli Configuration Manager

Note If a company has already implemented different data collection processes before Tivoli Data Warehouse installation, there is no need to modify them. The data manipulation phase is in charge of retrieving data from legacy sources and converting it into a common format.

Data manipulation (ETL1 and ETL2)

After the data sources are established, the next steps are:

- Loading legacy source data into a central data warehouse and transforming data into a standard format (ETL1 or Source ETL process).
- Selecting data subsets from the central data warehouse according to different business areas and loading them into data marts (ETL2 process or Target ETL process).
- Scheduling convenient time frames to run ETL processes.

Tivoli ETLs may be utilized immediately to manipulate data from Tivoli applications.

Data reporting

The final step of a Tivoli Data Warehouse process is producing timely updated reports according to the end user's specifications. Tivoli Data Warehouse provides a report interface that enables the creation of some basic reports against your data marts and their publication on an intranet or the Internet. You should notice that the report interface (RI) is not meant to replace OLAP or Business Intelligence (BI) tools. If your organization requires multidimensional reporting or a more sophisticated data analysis, you probably have to interface your data marts with external OLAP or BI tools, such as IBM DB2 OLAP, Crystal Decision, Brio Intelligence, Business Object, Cognos, SAS Institute, Microsoft OLAP Server, and so on.

23.4.1 Tivoli Data Warehouse in the enterprise

The architecture described in [Figure 23-5](#) on [page 587](#) shows one possible configuration. In this example, each location collects the event data and sends it to a centralized server.

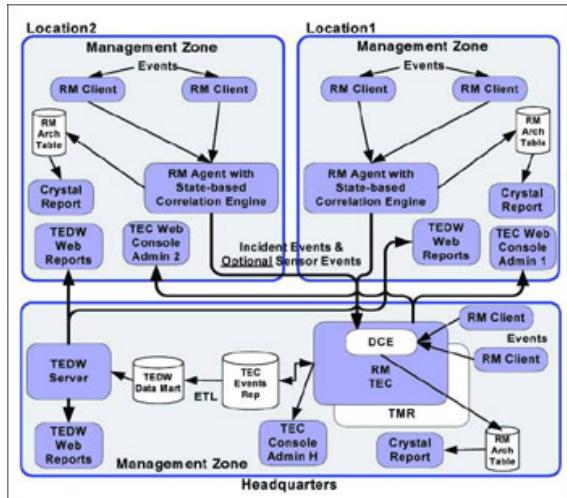


Figure 23-5: Tivoli Data Warehouse architecture

Each location has a reporting option from locally archived tables using Crystal Reports and using TEC event repository through Web access to the Tivoli Data Warehouse setup at headquarters. Headquarters administrators also have the option to set up JDBC access to the archive data at locations in order to generate and monitor location-specific reports as well. Individual administrators have access to their respective location security events as well as the enterprise security events. Super administrators at

the headquarters can manage all of the locations' security events and they have the option to present the global view of enterprise security events to the CIO.

23.4.2 Tivoli Data Warehouse in the company architecture

The Tivoli Data Warehouse architecture is built after the Risk Manager architecture, and all local events are stored in a local relational database. The Tivoli Data Warehouse server at headquarters uses the TEC event repository to analyze and generate enterprise-wide trends of security threat analysis.

Risk Manager 4.2 supplies these seven canned Tivoli Data Warehouse reports:

- 1. Service Compromise Events - Last 30 Days**

Shows the number of service compromise events (such as denial of service attacks) per day over the past 30 days.

- 2. Events by Destination Subnetwork - Last 30 Days**

Shows distribution of events over all of the subnetworks of the enterprise over the past 30 days. Two measures are displayed:

- Maximum events in a single hour over the time period
- Total events over the time period

- 3. Policy/Configuration Events - Last 30 Days**

Shows the number of policy, configuration, administrative, and state change events per day over the past 30 days.

- 4. Events by Destination and Category - Last 30 Days**

Ranks all hosts and event categories by the number of Tivoli Risk Manager events over the past 30 days.

- 5. Access/Authentication Events - Last 30 Days**

Shows the number of access and authentication events per day over the past 30 days.

- 6. Events by Destination Host - Last 30 Days**

Ranks all hosts by the number of Tivoli Risk Manager events over the past 30 days.

7. Infection Events - Last 30 Days

Shows the number of infection events (viruses, Trojan Horses) per day over the past 30 days.

You can customize these reports and create your own customized reports using the Tivoli Data Warehouse report interface. The following pages present some typical reports available through the Tivoli Data Warehouse enablement pack for Risk Manager.

Figure 23-6 depicts the 30-days trend for the events occurred over policy and configuration threats within the enterprise. This helps management to understand and improve policy change mechanisms by defining and configuring the change policy.

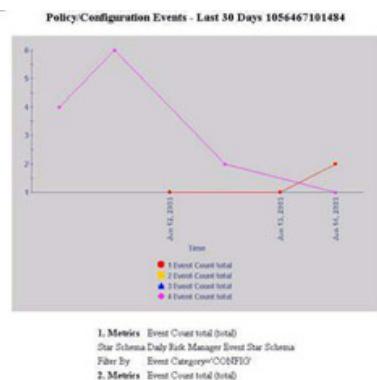


Figure 23-6: Policy / Configuration events, Last 30 Days

Figure 23-7 depicts the report for events by destination and host over the past 30 days. Management can analyze the events in order to take measures to increase and tighten security on those hosts.

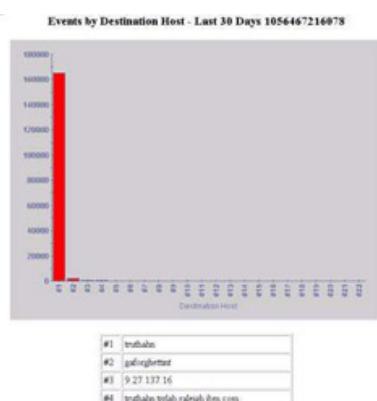


Figure 23-8 depicts the trend analysis over 30 days for virus infection events across the enterprise. This helps system and virus administrators to take preventive planning measures.



Figure 23-8: Infection Events, Last 30 Days

Figure 23-9 depicts the historical trend analysis for the security threats over 30 days by destination and by the threat category.

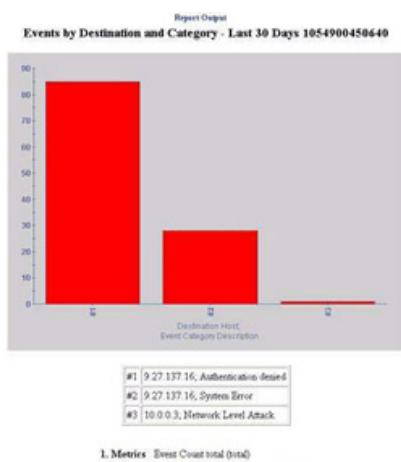


Figure 23-9: Events by Destination and Category, Last 30 Days

Figure 23-10 depicts the trend analysis for 30 days to assist the management in learning where the service is compromised over the security threats within the enterprise.

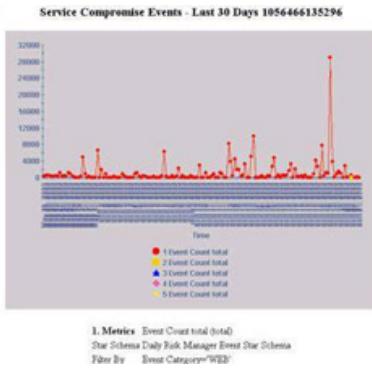


Figure 23-10: Service Compromise Events, Last 30 Days

23.4.3 Crystal Reports for Risk Manager

Tivoli Risk Manager contains 22 canned reports that were created using the

Compiled Reports feature of the Crystal Reports Designer program. The Compiled Reports format enables you to run reports without having the Crystal Reports Designer installed on the system. Windows is the only supported platform for installation of Risk Manager Crystal Reports.

Attention You must purchase and install the Crystal Reports Designer if you want to create or edit out-of-the-box Crystal Reports.

The compiled Crystal Reports provided with Tivoli Risk Manager include Firewall Management Reports, Intrusion Detection Reports, Risk Assessment Reports, Event Management Reports, and Virus Management Reports, described as follows.

Firewall management reports

The available reports for firewall management are:

1. Denied Connections by Network Address (rm_fw_02.exe):

This report summarizes the number of denied connections within the customer's network categorized by the type of connection denied.

2. Top Firewall Events (rm_fw_07.exe):

This report lists all classes of firewall-detected events, ranked by the number of events in each class and severity grouping.

Intrusion detection reports

The following reports are for intrusion detection:

1. Intrusion Events by Network Address (rm_ids_05.exe):

This report displays the number of IDS events categorized by event class for each host address. When the report is displayed, you may view details about each event.

2. Top Attacked Hosts (rm_ids_04.exe):

This report displays a bar chart showing the top 10 target hosts for intrusion events. You have the option to expand the data to display a pie chart showing the distribution of events by class.

3. Top Intruders (rm_ids_12.exe):

This report displays a bar chart showing the top 10 source hosts for intrusion events. You have the option to expand the data to display a pie chart showing the distribution of events by class.

4. Top Intrusion Events by Category (rm_ids_13.exe):

This report displays a bar chart showing the top 10 intrusion event categories. Examples of event categories are Web Attack, Service Attack, and IDS Level. You have the option to expand the data to display a pie chart showing the distribution of events by class.

5. Top Web Intrusion Events (rm_ids_14.exe):

This report displays a bar chart showing all classes of Web intrusion events (where sensor type is Web IDS), ranked by the number of events in each class.

Risk assessment reports

The following reports are for risk assessment:

1. Access Violations by Network Address (rm_ra_01.exe):

This report summarizes denied access attempts within the subnetwork categorized by the class of denied access.

2. Authentication Events by Network Address (rm_ra_02.exe):

This report displays the number of authentication events for each host address. A summary chart at the beginning of the report shows the hosts with the most authentication events.

Authentication events are broken down into two broad categories: Allowed and Denied. You have the option to double-click an

authentication category to view a list of individual events for the selected host and category.

3. Failed Login Attempts by Host and User (rm_ra_03.exe):

This report displays failed login attempts by host. You have the option to use the Group By parameter to view the data from the point of view of:

- a. Destination hosts (what machines are being logged onto)
- b. Source hosts (what machines are being logged on from)

A summary chart at the beginning of the report displays the top 10 hosts ranked by number of failed login attempts. The report text lists all hosts, ranked by number of failed login attempts. If you double-click on a host in the text section, you have the option to see a pie chart showing the breakdown by user ID. If you double-click on a slice of the pie chart, you have the option to see details about each individual event.

Event management reports

The following reports are for event management:

1. Events by Category (rm_evt_08.exe):

This report summarizes all events by category. Examples of category names include Denial of Service, Network Level Attack, and Trojan Horse.

2. Events by Customer (rm_evt_04.exe):

This report displays a pie chart showing the distribution of events by customer name. Customer name may be a company name, line of business, geography, department number, or any customer-specific ID. You have the option to double-click on a customer name to see a bar chart showing the distribution of events by host and class category. Double-click on the desired segment of the bar chart to see a listing of individual events within each host and class category.

3. Events by Month (rm_evt_02.exe):

This report displays a summary chart showing the number of events received each month, starting with the beginning of the current year. You have the option to also select last year or the year before, using the Year parameter. You have option to view

weekly event summaries, by double-clicking on any month in the chart. From the weekly summary chart, double-click on any week to see event summaries by day. To view a chronological listing of all events received for a particular day, double-click on the bar for that day.

4. Events by Severity (rm_evt_03.exe):

This report displays a pie chart showing the distribution of events by severity. You can double-click on a severity level to see another pie chart showing the distribution of events by class. To see a listing of individual events within each class, double-click on the desired class.

5. Events for the Last 7 Days (rm_evt_07.exe):

This report displays a bar chart showing the number of events for each day of the past 7 days. Each colored segment on a bar represents the number of events for a particular sensor type group (see *Events per Day by Address and Sensor Types* below for a description of sensor type groups). You can double-click on a colored segment to display an event list for a single day and sensor type group.

6. Events for the Last 24 Hours (rm_evt_24.exe):

This report displays a chart showing the total events by hour over the past 24 hours, broken down by sensor type groups. Each colored segment on a bar represents a sensor type group (see *Events per Day by Address and Sensor Types* below for a description of sensor type groups). You can double-click on a segment to display an event list for a single hour and sensor type group.

7. Events per day by Address and Sensor Type (rm_evt_05.exe):

This report displays the distribution of events by sensor type across a network. A summary 3-D chart shows the top seven IP addresses with the most events, categorized by sensor type groups. The text section shows the number of events received per day at each host in the network, categorized by sensor type groups. All events originate from one of the following sensor type groups:

- Antivirus: Sensors such as Norton and McAfee.

- Database: Sensors that report on security-related access attempts and configuration changes to database management systems (DBMS) such as Oracle and DB2.
- Firewall: Sensors such as Check Point FireWall-1, Cisco Secure PIX, IBM Firewall, and ZoneAlarm.
- Host IDS: Host-based sensors such as AIX, Solaris, Linux, Windows, and RealSecure System Agent.
- Network IDS: Network-based sensors such as NIDS, Cisco Router, Cisco Secure IDS, RealSecure IDS, and SNORT.
- Web IDS: Web-based sensors such as Web Intrusion Detection System (Web IDS).
- Wireless: Wireless-based sensors such as Wireless Security Auditor.

8. Event Query (rm_evt_10.exe):

This report displays the distribution of events by user selectable groups. You must select a Primary and Secondary group parameter from the following list:

- Destination Host
- Source Host
- Customers
- Sensor Type
- Sensor Host
- Event Category - Event Class
- Severity
- Date
- Day of Week
- Hour of Day

The default for Primary group is Customer; the default for Secondary group is Destination Host. The report displays a summary bar chart showing the top 10 event counts within the Primary group. Double-click on a bar to see a pie chart showing

the top 15 event counts within the Secondary group. Double-click on a slice in the pie chart to see a listing of all events in the selected group.

9. Top Events (rm_evt_09.exe):

This report lists all classes of events by the number of events in each class and severity grouping.

Virus management reports

The following reports are for virus management:

1. Antivirus Scans and Updates (rm_av_10.exe):

This report displays all hosts that have had virus scans and signature updates, ranked by number of days since the last virus scan. The maximum number of days counted is 30. If a host has not been scanned for viruses in the past 30 days, then it will be assigned a value of 30 days since the last scan. A summary bar chart shows the top 10 hosts that have not been scanned recently. For each host, one bar shows the number of days since the last scan and another bar shows the number of days since the last update. In the text section of the report, you can double-click on a host to see a chronological list of virus scans and signature updates for that host.

2. Top Infected Hosts (rm_av_06.exe):

This report displays a bar chart showing the top 10 hosts infected with viruses. Double-click on a host to see a list of viruses with date, time, antivirus vendor name, and virus description.

3. Top Viruses (rm_av_02.exe):

This report displays a pie chart showing the top 15 virus signatures. You can double-click on a signature group to view virus event details, including time of event, host affected, antivirus vendor name, and a descriptive message.

Some of the sample Crystal Reports are presented on the following pages. These reports help you to understand the design of out-of-the-box Crystal Reports for Risk Manager.

[Figure 23-11](#) depicts the Web Intrusion events that help Web administrators to take necessary preventive measures. Textual details are available as follows.

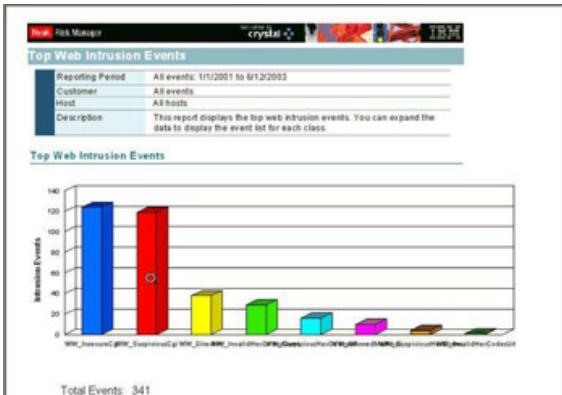


Figure 23-11: Top Web Intrusion Events

Figure 23-12 depicts the threats or intrusions occurrences on Web servers in textual form. This helps Web administrators to take necessary action and update the firewall administrator to take preventive measures.

Event Time	Source / Target Host	Description	Number of Events
2003-06-11 23:38:22	8.188.128.180 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 17:46:07	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 15:00:53	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 14:16:16	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 06:06:29	9.14.3.112.69 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 05:08:14	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 04:59:20	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 04:28:56	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 04:28:16	9.188.107.233 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 04:25:04	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-11 01:27:25	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 23:48:12	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 23:36:36	9.27.44.141 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 18:51:23	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 18:49:16	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 15:50:29	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 14:48:36	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 14:13:02	9.27.84.199 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-10 11:14:16	9.27.25.55 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 19:36:41	9.185.65.64 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 18:29:17	9.14.2.163 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 17:47:24	9.36.233.16 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 17:32:00	9.27.84.39 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 15:18:51	9.27.82.23 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 14:34:13	9.27.83.39 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-09 04:23:37	9.188.161.167 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-08 05:28:11	9.42.201.207 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-07 14:15:26	9.18.34.154 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-07 13:37:31	9.36.233.16 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-07 12:16:15	9.14.8.5.142 / galaphettet	Suspicious cgi.CodeRedII	1
2003-06-07 12:16:09	9.14.8.5.142 / galaphettet	Suspicious cgi.CodeRedII	1

Figure 23-12: Top Web Intrusion Events, text format

Figure 23-13 depicts the NIDS events by network address. This helps network administrators to facilitate firewall configuration to bar those network addresses on the firewall.



Figure 23-13: Intrusion Events by Network Address

Figure 23-14 depicts the top firewall events over the location. This helps firewall administrators to take preventive measures based on the events. It is available in textual form, to drill down to the event details.

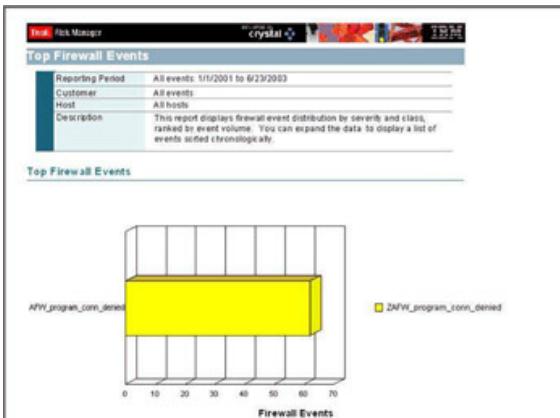


Figure 23-14: Top Firewall Events

Figure 23-15 depicts the top events occurrence for a location, which helps the administrator to plan the remedial actions in advance.



Figure 23-15: Top events

Figure 23-16 depicts the details of top events in textual form to show more details.

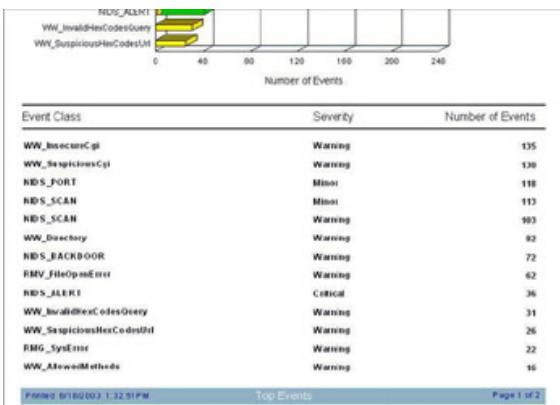


Figure 23-16: Top Events text report

Figure 23-17 depicts the events per day by address and sensor type. This helps administrators to manage and fine tune the sensors.

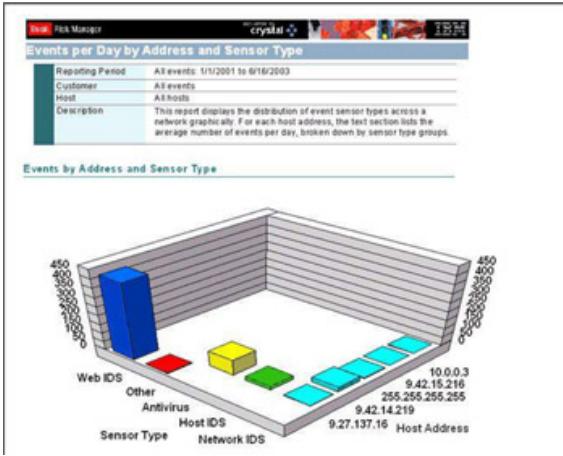


Figure 23-17: Events per Day by Address and Sensor Type

Tivoli Data Warehouse and Crystal Reports provides a multitude of other views. You may also filter the reports to provide separate views for the individual companies or partners in your enterprise. Reports can be built for each location and for the entire company. Tivoli Data Warehouse and Crystal Reports provide a Web-based access method that gives you the ability to produce reports and enable users to review reports only for their environment with a Web browser.

23.4.4 Mapping the solution to the organization

The organizational flow design shown in [Figure 23-18 on page 605](#) provides help to everybody involved within the enterprise to organize security management.

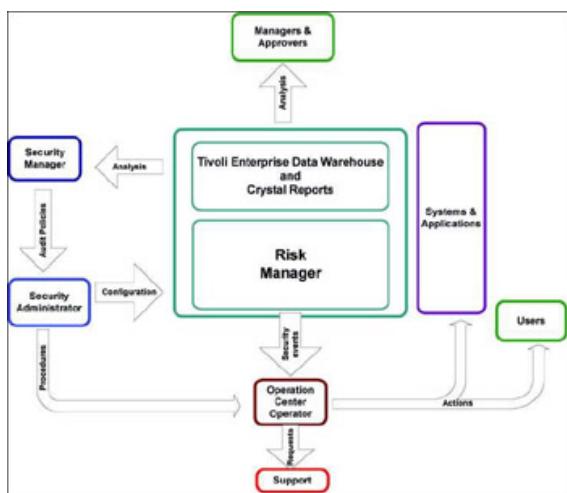


Figure 23-18: Organization flow

- Managers: They consult the reports coming from Tivoli Data Warehouse and Crystal Reports, which help them supervise activities within the enterprise.
 - Security Manager or Security Officer: Responsible for defining the audit policies (what, when, and how). To refine these policies, the manager uses Tivoli Data Warehouse and Crystal Reports to figure out the strengths and weaknesses of the overall implementation.
 - Security Administrator: Physically configures the Risk Manager files (thresholds, categories, adapters, and so on) to implement the audit policies defined by the Security Manager or Security Officer. Another administrator task is to document instructions that describe incidents, what actions should be taken for specific events, and to build automated scripts when it is possible. This is an ongoing task as new threats are discovered and new tasks are needed to protect the network.
 - Operators: Operators sit in front of the console to receive security audit events. Their job is to react to incidents by executing the procedures and documents written by the Security Administrator. They interact with the System administrators, Application administrators, or both, as well as with the users in order to solve the problem. For continuity of service, they have to be organized to support 24x7 availability.
 - Support: This function can be implemented as an external product support or internal customer and user support. The aim is to assist operators to solve the problem by performing tasks the operator is not authorized to do.
-

23.5 Summary

Tivoli Risk Manager, along with Tivoli Data Warehouse, provides a flexible and scalable tool for consolidating security alert monitoring. Configurations are implemented to meet a company's geographic, staffing, and organizational policies.

In addition to the real-time alert management, Risk Manager provides decision support capability to review the historical data collected and project trends and potential issues. The application helps with the identification of threats and security breaches, and offers best practices actions for mitigation and remedy.

In this chapter we have seen how to meet the challenge of integrating separate entities into a secure corporate environment after a merger, using consistent security policies and management processes.

Part 5: Using MASS in Business Scenarios

Chapter List

[Chapter 24:](#) Global MASS: An Example

In this part, we discuss how to apply the Method for Architecting Secure Solutions (MASS) to specific business scenarios in order to define an overall enterprise security architecture.

Chapter 24: Global MASS: An Example

In our final chapter, we present a practical example of applying the Method for Architecting Secure Solutions (MASS) within an e-business IT infrastructure.

As described in [Chapter 2, “Method for Architecting Secure Solutions”](#) on [page 15](#), 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 discusses the four steps of this process.

24.1 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.

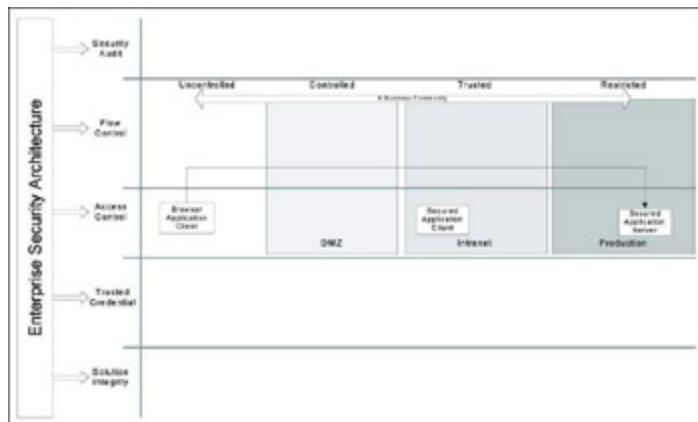


Figure 24-1: Business view

The view depicted in [Figure 24-1](#) 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.

24.2 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 24-2](#).

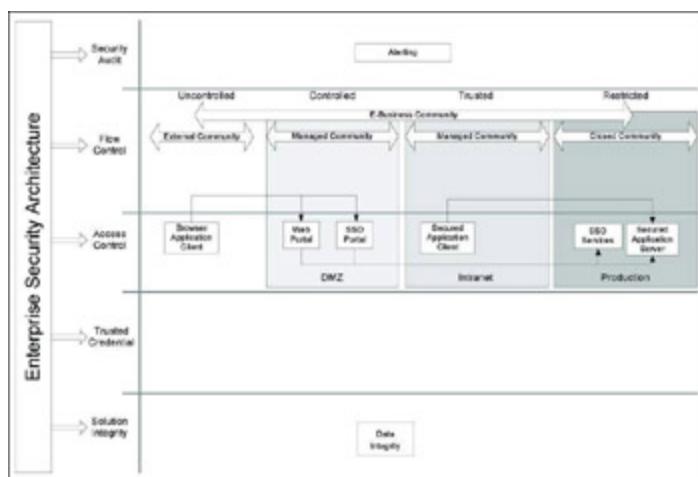


Figure 24-2: Logical view

24.3 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 24-3](#) on [page 613](#) draws the picture.

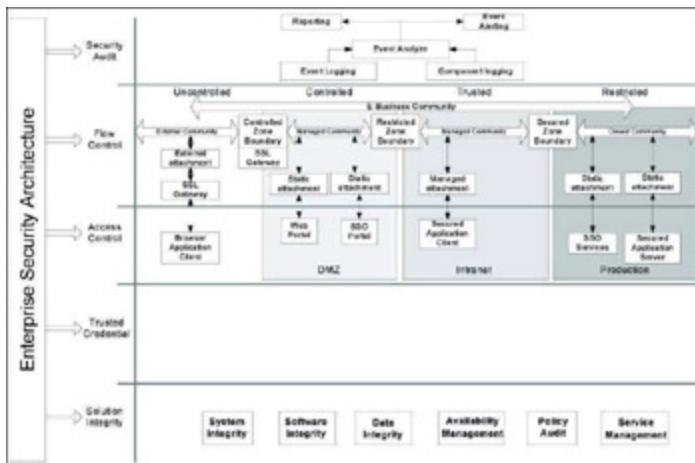


Figure 24-3: Detailed view

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).

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.

24.4 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 24-4 on page 615](#).

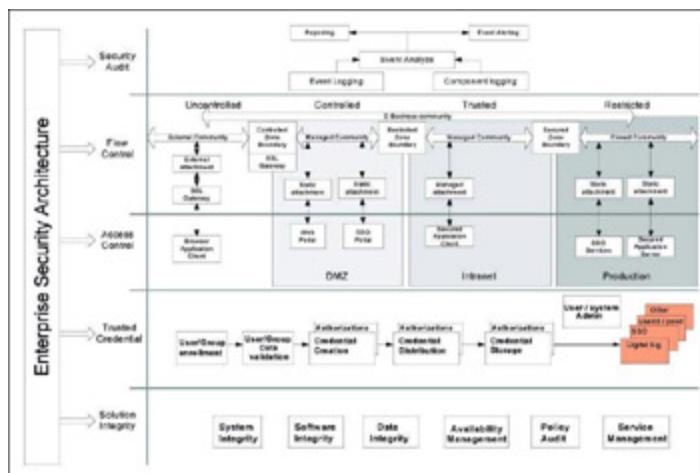


Figure 24-4: The full architectural view

Part 6: Appendices

Chapter List

[Appendix A:](#) Additional Product Information

[Appendix B:](#) Single Sign-On - A Contrarian View

Appendix A: Additional Product Information

Overview

This appendix provides additional product information for firewalls, PKI, and virus protection software that were not discussed in [Chapter 3, “IT infrastructure topologies and components” on page 49](#).

This appendix contains information about:

- Firewalls
 - Cisco PIX
 - Symantec Enterprise Firewall
 - Check Point VPN-1/FireWall-1
 - PKI environments
 - RSA Keon
 - Verisign
 - Antivirus packages
 - Norton AntiVirus
 - Panda Global Virus Insurance
 - Trend AntiVirus
-

Firewall packages

Firewalls are part of the security architecture of a network. They protect resources from other networks and individuals by controlling access to the network and enforcing a security policy that can be tailored to suit the needs of the enterprise. There are many firewalls available for purchase today. This section mentions three: Cisco PIX, Symantec Enterprise Firewall (formerly Axent Technologies Raptor Firewall) and Check Point Software's Check Point Firewall-1.

Cisco PIX

The Cisco PIX Firewall solution is an integrated hardware/software appliance that uses a version of the Cisco IOS and offers good performance. The product line is available for almost any size of operation or enterprise. Cisco's PIX firewalls provide the latest in security technology, including stateful inspection, IPSec, L2TP/PPTP based VPNs, content filtering, and integrated intrusion detection.

The center of the PIX firewall family is an Adaptive Security Algorithm (ASA), which maintains secure perimeters between networks controlled by the firewall. The ASA design creates session flows based on source and destination TCP sequence numbers, port numbers, and additional TCP flags. Some models of Cisco PIX firewalls enable you to create multiple demilitarized zones (DMZs) for your network by adding additional interface cards.

Cisco PIX firewalls provide a scalable and secure architecture. They also offer reliability by leveraging integrated stateful failover capabilities that can allow network traffic to be sent to a hot standby automatically in the event of a failure, supported by maintaining concurrent

connections via automated state synchronization between the primary and the secondary. PIX also provides Network Address Translation (NAT) and Port Address Translation (PAT) to concealed IP addresses of internal networks and to expand network address space for internal networks.

For more information about this and other Cisco products, visit:

<http://www.cisco.com>

Symantec Enterprise Firewall

Symantec Enterprise Firewall (formerly Axent Technologies Raptor Firewall) is a full-featured security software package that enables you to protect your network from outside threats. Enterprise Firewall is fairly easy to install and configure. It includes such features as content filtering, Out of Brand Authentication, Windows NT Domain Authentication, and RSA SecurID or CryptoCard. Combined with the optional packages Symantec Enterprise VPN (formerly PowerVPN) and personal firewall, Symantec Enterprise Firewall can also extend the corporate perimeter to provide secure, low-cost connectivity for remote offices and telecommuters.

Symantec Enterprise Firewall minimizes your network's vulnerabilities and delivers full-featured security, offering control of information entering and leaving the enterprise but still providing partners and customers with secure access to resources. Its support for user authentication methods such as Radius, Digital Certificates, LDAP, and NT domain authentication gives administrators the flexibility to use existing security policies. Symantec Enterprise Firewall offers both hardware-based and software-based solutions as well as integrated Web and Usenet content filtering.

Developed for the Windows NT/2000 and Solaris platforms, Enterprise Firewall enables administrators to manage security policies from a central console. Enterprise Firewall can be utilized as a basic communication mechanism with network devices deployed with a DMZ or with a DMZ and multiple firewalls. With the proper licensing, you may deploy Enterprise Firewall wherever you perceive a weakness in your network.

For more information visit the Symantec Web site at:

<http://www.symantec.com>

Check Point Software's Check Point FireWall-1

Check Point FireWall-1 is a full-featured enterprise security product that has a modular and scalable architecture. This component approach enables you to design and implement security as your business needs change and to manage those changes from a central point. As with the preceding products, console management, authentication, security policy implementation, and LDAP are core components. However, with the addition of the many optional modules available, you may expand the breadth of the base product.

FireWall-1 software is designed to run on Windows NT, Sun Solaris, Red Hat Linux, HP-UX, and IBM AIX.

FireWall-1 supported hardware platforms include Check Point VPN-1 Appliances, ODS SecurCom 8000 family, Alcatel (Xylan) switches, Nortel ARN - ASN - BN System 5000 routers, and Nortel Connectivity switches. FireWall-1 is designed to integrate with the following network interfaces: ATM, Ethernet, Fast Ethernet, FDDI, and Token Ring.

PKI

RSA Keon

For much more detailed information concerning FireWall-1 and its various component packages, visit Check Point Software's Web site at:

<http://www.checkpoint.com>

X.509 Public Key Infrastructure (PKI) is an Internet Official Protocol Standard that is, in simple terms, a digital ID. It is used to verify that a message or document was authored by a certain person, and that it was not altered or modified by anyone else. Digital IDs are a standard way to establish proof of identity using a variety of protocols. They certify that a document was signed by a person with a certain public key when opened by the recipient that holds the private key. It does not matter whether someone else holds the original public key, because it is of no value in decrypting the document.

The discussion regarding how this is achieved is lengthy and beyond the scope of this redbook. More detailed information may be found in the *RFC2459 Internet X.509 Public Key Infrastructure* at <http://www.ietf.org/rfc.html>.

RSA offers a family of PKI products for e-business that alone or together offer the ability to manage digital certificates.

- RSA Keon Certificate Authority

RSA Keon Certificate Authority software issues, manages, and validates digital certificates. The software includes secure administration, enrollment, and directory and logging servers, as well as a Simple Certificate Enrollment Protocol (SCEP) server

that provides automatic enrollment for issuing certificates to SCEP-compliant Virtual Private Network (VPN) devices. RSA Keon Certificate Authority software also features a powerful signing engine for digitally signing end-user certificates and system events, as well as an integrated data repository for storing certificates, system data, and certificate status information.

- Registration Authority

RSA Keon Registration Authority (RA) software works with the Certificate Authority (CA) to streamline the enrollment process for handling large volumes of end-user certificate requests. It verifies the credentials of certificate requests and provides certificates to the requestor. The RSA Keon RA software enables organizations to set up either remote or local stand-alone enrollment centers for large user implementations at distributed geographic locations, thereby allowing the organization to scale its certificate management system while moving the approval process closer to the users.

- Key Recovery Module

The RSA Keon Key Recovery Module (KRM) software, an optional component of the RSA Keon CA software, provides a way to securely archive and recover users' private encryption keys. It combines reliable and secure long-term encryption key-pair storage with straightforward, secure user enrollment.

For more information, see:

<http://www.rsasecurity.com/products/keon/index.html>

VeriSign

VeriSign offers several different options for PKI as well as other services that are outsourced.

OnSite Managed Trust Service

VeriSign OnSite is a fully integrated PKI managed service designed to secure intranet, extranet, Virtual Private Network (VPN), and e-commerce applications by providing maximum flexibility, performance, and scalability with the highest availability and security.

The OnSite service enables you to establish a customized PKI and Certificate Authority (CA) system for issuing digital certificates throughout your enterprise. Unlike software-only solutions or building your own PKI, the OnSite managed service enables you to leverage the knowledge at VeriSign without adding additional tasks to your internal staff. With OnSite, you control certificate issuance and management while VeriSign provides a backbone of certificate processing services.

For more information, visit:

<http://www.verisign.com/products/pki/index.html>

Antivirus software

Why include antivirus software in a discussion of security architectures? Viruses are a breach of your system, can result in lost productivity and down time, and ultimately lost revenue. Among the most important ways to protect your online assets should be the distribution and use of antivirus software.

Packages available today are easy to install and have an effective update mechanism.

Norton AntiVirus by Symantec

One of the world's leading antivirus solutions delivers advanced virus detection and elimination software to businesses of all sizes. This multiplatform solution provides easy-to-use, customizable, install-and-forget protection that begins guarding your workstations, servers, and gateways as soon as it is installed.

Safety first

Rare or previously unknown viruses do not have to be fatal. During installation, Norton AntiVirus/IBM® Solution Suite provides a generic disinfection feature, a patent-pending technology that records information about your files for later use in disinfection. If a file is infected with a rare or previously unknown virus, generic disinfection attempts to use the recorded information to reconstruct the original uninfected file. If the reconstruction exactly matches the original file, the file will be safely disinfected.

Automatic updates

With an average of 10 to 15 new viruses discovered every day, it is critical to keep virus protection current. LiveUpdate enables automatic retrieval of new virus definitions from Symantec up to once a week. Run your first Live Update session during installation and schedule future sessions to run automatically, helping ensure that your network is always protected against the latest virus threats without interrupting network activity.

Efficient performance

Exclusive Scan Caching technology provides fast scanning time and low bandwidth consumption. After a file has been scanned and found to be virus-free, it will not be scanned again until changes are made, greatly improving day-to-day server performance. Norton AntiVirus scans all files in real time as they are accessed or copied to and from the server. You can save even more time by scheduling automatic scans at regular intervals.

Quarantine infected files

Norton AntiVirus Quarantine technology provides a safe zone on a Quarantine server where you can isolate infected files and attachments for repair when network activity is lower. This quarantine area helps to ensure that your other files stay clean and that you do not accidentally send infected files to anyone else. Quarantines, virus scans, and removal can be managed from any location using a Web browser.

Product Highlights

- Detects and removes known, previously unknown and unidentified viruses Works in the background to provide continuous, automatic protection Provides broad coverage of clients, servers, and gateways

- Detects and disinfects most viruses
- Analyzes viruses byte-for-byte before disinfection
- Offers the same look-and-feel across multiple platforms
- Automates and centralizes administration
- Virtually eliminates false alarms
- Delivers automated, server-based updates of the AntiVirus products

For more information, visit
<http://www.symantec.com/product/>.

Panda Software Global Virus Insurance

Global Virus Insurance is an easy-to-use, effective package that includes the Panda Administrator. Global Virus Insurance offers protection for NT Servers, Exchange Servers, Novell Servers, and all workstation platforms, such as Windows 95/98/ME/XP, Windows NT Workstations, Windows 3.1x, MS-DOS, and OS/2®.

Centralized deployment across all workstations, servers, and e-mail platforms is simplified with technology that scans at the Winsock and TCP/IP level. One signature file fits all platforms. Panda Software's Virus Signature Database can be updated every day and the package can be configured to download the update automatically. When the download is complete, Panda Administrator automatically sends the update to all workstations to ensure that the updates are installed on all workstations.

Product features

- Daily, automatic, intelligent updates of the Virus Signature Database.
- Automatic distribution of updates to all workstations on your network.
- Scans and disinfects the FTP, NNTP, POP3, and SMTP Internet protocols at the Winsock level.
- Offers scanning and disinfection for all of the most widely used e-mail clients in the market, both MAPI and/or POP3. Also integrates into Exchange and Notes/Domino GroupWare environments.
- The only antivirus product that offers scanning and disinfection in the body of e-mail messages, both in Rich Text Format (RTF) and HTML-formatted e-mail messages. Panda also scans and disinfects OLE embedded objects and nested messages, as many levels as needed.
- Performs e-mail scanning while in memory, as opposed to copying files to temporary drives, disinfecting them with the desktop antivirus, and re-attaching them to e-mail messages.
- Free 24-hour S.O.S. Virus service.
- Centralized antivirus management for the corporate and enterprise infrastructure, with end-to-end protection for the widely used operating system platforms.
- MAPI-Compliant antivirus that scans and disinfects the Message Transfer Agent (MTA) Connectors, such as the Internet Mail Connector and the X.400 Connectors.

For more information, refer to the Web site:

<http://www.pandasecurity.com/gviinfo.htm>

Trend Antivirus Solutions

Trend Micro, Inc. is a leader in network antivirus and Internet security software and services over a large variety of packages for your enterprise. Its solutions protect the flow of information on PCs, file servers, and e-mail servers and at the Internet gateway, providing a complete, centrally controlled VirusWall for enterprise networks. With an innovative and technology-driven focus, Trend Micro has migrated antivirus applications from the desktop to the network server and the Internet gateway with ease of installation and use, multi-platform support, and strong central management tools.

For a complete set of product descriptions and management tools, visit:

<http://www.trendmicro.com/en/products/global/enterprise.htm>

Additional resources and information

For more information about virus-related issues or antivirus products, visit these Web sites:

- CNET Networks: <http://www.cnet.com>
- Ziff-Davis Press: <http://www.zdnet.com>
- Consumer Information Organization: <http://Anti-Virus.com>
- Carnegie Mellon Software Engineering Institute: <http://www.cert.org>

- Computer Security Resource Center:
<http://csrc.ncsl.nist.gov>
-

Appendix B: Single Sign-On - A Contrarian View

The purpose of this appendix is to outline some of the issues associated with achieving Single Sign-On (SSO) in a real enterprise environment. Many organizations embark on these efforts without having thoroughly considered the issues and, more important, the costs and benefits. We hope that this discussion will help explain the issues and the implications of an SSO solution. The content of this appendix is the work of Keys Botzum, a Senior Consulting I/T Specialist in the IBM Software Services, WebSphere Practice.

Because we have not modified the original content of the article, some of the product-related information may be out of date. This is, however, not relevant to the overall discussion.

Introduction

Single Sign-On (SSO) is the holy grail of many organizations. Broadly, by achieving SSO, users will log in once to an SSO domain and then never be challenged again while accessing multiple secured resources. The Open Group defines SSO as:

Single Sign-On (SSO) is 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]

Originally, this was to be achieved by developing all applications and tools to use a common security infrastructure and avoid the current situation of heterogeneous security infrastructures. This includes a common format for representing authentication information or credentials. This approach implies a number of valuable benefits:

- For end users
 - Only one authentication mechanism to remember. For password-based authentication, this means users only have to remember one password.
 - If using passwords, users only have to update one password and follow one set of password rules.
 - A single sign-on for each user into the SSO domain, typically only once per day.

- For I/T operations
 - A single common registry of user information (possibly replicated).
 - A single common way to manage user information.
 - A single common security infrastructure.
- Security advantages
 - A common secure infrastructure leveraged enterprise wide that can be carefully managed and secured.
 - Depending on the security approach taken, secure delegation of credentials is possible. This will enable end-to-end security, possibly across application and system boundaries.
 - Easier to manage and protect common registry.
 - Easier to verify user security information and update when necessary rather than tracking down all operational systems. This is particularly valuable when users move to new roles with different access levels.
 - Can enforce common enterprise wide password and security policies.
 - Users less likely to write down passwords since they only have to remember one.

[1] Taken from the security section of the Open Group Web site (<http://www.opengroup.org/security/topics.htm>)

The problem

Creating a common enterprise security infrastructure to replace a heterogeneous infrastructure is, without question, the best technical approach. This has been and is being attempted with technologies like the OSF Distributed Computing Environment (DCE), Kerberos, and with PKI based systems, but few, if any, enterprises have actually achieved this.

Unfortunately, the task of changing all existing applications to use a common security infrastructure is very difficult. This has been further hampered by the lack of consensus as to what should be the common security infrastructure. Today we have many proprietary security systems, as well as several competing security standards and pseudo-standards: Kerberos, DCE, Microsoft Active Directory and related technologies, PKI, and so on. Of course, these technologies do not interoperate seamlessly. As a result, these proprietary and standards based solutions, while very appropriate, cannot be applied to every system. For example, until recently^[2], WebSphere Application Server could not recognize authentication information from systems other than itself. Organizations that had some proprietary Web based authentication system found that they were problematic with WebSphere Application Server. Problems like these occur over and over again across an enterprise as organizations buy products without fully considering the implications.

[2]WebSphere Application Server now (as of WebSphere Application Server 3.5.3 and later) supports a feature known as Trusted Associations that allows the WebSphere Application Server security infrastructure to recognize (via custom coding) credentials from other security systems

when provided to Web based applications. The Lightweight Third-Party Authentication (LTPA) cookie provides another means of forwarding user's credentials.

The Proxy-SSO “Solution”

Since moving all applications to a single common security infrastructure is extraordinarily difficult, many enterprises attempt to create the illusion (for end users) of one common security infrastructure through the use of a number of SSO products that support some mechanism to authenticate to existing legacy systems without user intervention.

Throughout this document I will refer to this as proxy-SSO to distinguish this from *true* SSO achieved by creating a common security infrastructure. These products typically use some proprietary authentication mechanism and then reauthenticate transparently to multiple underlying systems via the user interfaces to those systems. This involves some form of scripting engine that drives the interaction with each underlying system’s authentication challenge. The scripting engine simulates a user logging in like normal so the underlying systems do not need to be changed. A proxy-SSO database stores all combinations of user names, passwords, and systems.

The solution is difficult, expensive, and incomplete

The task of configuring a proxy-SSO system for a typical large enterprise with numerous existing systems is staggering, as custom scripts must be developed for every possible user authentication interaction. For the UNIX operating system utilities, we have ftp, telnet, rsh, and rlogin. Add to this list the utilities for other operating systems, custom applications, and Web sites, and the number is enormous. In addition, once the scripts are created, they must be maintained. As the underlying systems change their interfaces, the scripts will require changes. This results in an ongoing maintenance cost that is rarely considered. The scripting also hides the “front-door” to an application from the end users and thus might bypass important information such as security warnings that are displayed on the login page.

A central proxy-SSO database of user name and password mappings must also be created and maintained. This database must follow all password restriction and invalidation rules for all enterprise systems. It must be updated whenever underlying passwords are changed, or it must be the only system that changes passwords (creating yet another scripting task for each enterprise system). This does not even address systems that use other authentication mechanisms, such as certificates, challenge-response, SecureID cards, and so on.

Here is a summary of the costs of proxy-SSO:

- Initial costs
 - Product purchase.

- Customization of product for existing systems. This usually involves creating custom scripts to drive the legacy systems. This is a large work effort.
 - Loading existing user information into the proxy-SSO solution and deploying to users.
- Ongoing costs
 - The usual software upgrade costs.
 - Another registry to maintain. This system must be highly available.
 - Password management by users. When passwords expire in the various legacy systems, users must update the legacy system and the SSO system. Since users are no longer familiar with the login procedure for the legacy systems, they may not even know their passwords. This may make it impossible for users to change their own passwords without assistance.
 - Script maintenance. As the legacy system user interfaces change, the scripts will have to change. It is important to understand how much effort will be required to do this.

Of course, many of these costs can be controlled by carefully designing the underlying applications to work with a script-based proxy-SSO solution, but this largely defeats the main tenant of these proxy-SSO solutions: no need to modify “legacy” systems.

On the benefit side, the proxy-SSO solution saves users the trouble of typing in their user name and password to several

different systems several times per day and remembering all of these user name and password combinations. While this is certainly an issue, it is not clear how much real benefit this provides to the organization. The most troublesome aspect of multiple authentication systems is probably remembering the number of passwords, not the actual authentication effort. This point will be revisited later. Of course, in very high cost situations, such as call centers, these benefits are significant and can probably justify the cost of a SSO solution. In other, more routine environments, the benefit is far less clear.

From a security perspective, the results are mixed. The enterprise still has multiple incompatible security infrastructures and now has one more security product and one more user registry (another point of attack). On the positive side, if the proxy-SSO solution can manage the changing of user passwords in all legacy systems transparently, then users will no longer have difficulty remembering passwords and will no longer write them down.

In summary, proxy-SSO solutions tend to provide limited benefit to end users while not addressing most of the operational and security costs of a heterogeneous security infrastructure and introduce new operational costs.

The Web

Recently, efforts to achieve fully general SSO have largely died out. Now, however, a newer form of limited SSO is becoming popular: Web based SSO. In this space, there are products that authenticate users using some mechanism and then create credentials that can be used by downstream Web applications for authorization. This is the perfect insertion point of a Policy Director based Web portal that can provide true Web SSO to all back-end Web application spaces.

Unfortunately, as in the non-Web space, changing all existing Web systems is difficult. Thus, people are again embarking on proxy-SSO solutions for the Web. It is believed that this is simpler because Web applications have just one interface: the Web browser. However, while a single user interface simplifies the problem, it does not mean that all Web applications authenticate the same way. There are many different ways of asking a user to authenticate. Even in a simple Web based application, the proxy-SSO scripting engine must be able to traverse an arbitrary Web application and enter input just as if the user had done so. This problem, while significantly simpler than enterprise proxy-SSO, is still difficult in general.

To script a Web interaction around a password based authentication system, the following issues must be considered:

- The initial request to the target site must be rerouted to the proxy-SSO server that will then retarget the request appropriately.
- Basic authentication is trivial, because the challenge protocol is well defined and easy to intercept.

- Form based authentication presents the following unique problems:
 - The login page must be loaded and parsed in order to extract the needed hidden fields and find the action and target servlet.
 - Forms may ask additional questions beyond user name and password.
 - There may be more than one action on the form.
 - Sites may use elaborate Java Script to challenge the user and/or encrypt the user's authentication information before sending it back to the server.
 - Browsers are notoriously lenient in their presentation of illegal HTML. The proxy-SSO solution needs to work with illegal HTML that is browser displayable to avoid forcing changes to existing sites.
- Traversing a Web site to simulate a user login interaction requires addressing these issues:
 - Some sites present informative messages to users before the login process starts. Bypassing these messages, particularly if they are security warnings, is problematic.
 - Cookies sent before the post-login welcome page of the target application must be preserved.
 - Sites that detect the browser type and make decisions before or during the login process

must be handled. This information must be passed through from the browser by the proxy-SSO solution.

- Sites may create dynamic URL strings during and before the login process with encoded information in the URL (for example, [http://www.abc.com/index.html?
session=12345](http://www.abc.com/index.html?session=12345)).
- Sites may pop up additional browser panels during the login process
- Proxy server (in the Web proxy server sense) solutions must address these additional issues:
 - URLs embedded in the responses must be rewritten to point to the proxy. Embedded URLs may be absolute (for example, <http://www.abc.com/app/page.html>), server root relative (for example, /app/page.html), or page relative (for example, page.html).
 - URLs embedded in Java Script must be rewritten. For some systems, this may be impossible if the URLs are computed on the client. Even if not, this requires parsing the entire response stream looking for particular characters. This will affect performance.
 - Cookies sent back by the site must have the Domain and Path values rewritten to point to the proxy so the browser will not discard them.
 - Multiple sites, when combined under a single proxy, may reference resources that conflict

with each other. URLs and cookie names may no longer be unique.

It is worth noting that some applications that appear to be Web applications are so sufficiently complex that they are really full-fledged applications. Many “Web” applications use plug-ins, extensive Java Script, and Java applets to create a fuller user experience. These types of applications may not be addressable via the techniques described above.

One should not forget that Web application user interfaces change rapidly and this will affect scripts that depend on the existing interface. It should be obvious, that while the Web solution simplifies the client interaction problems, it is hardly trivial. Finally, the other operational issues (heterogeneous infrastructure, multiple, registries, password maintenance, and so on) associated with a proxy-SSO solution remain.

An alternative approach

We have spent quite a bit of time discussing the weaknesses of the proxy-SSO approaches. Now it is time to look at the problem more closely and try to determine an approach that achieves as many of the benefits of “true” SSO as possible without creating new operational problems.

Looking back at the benefits of SSO from [“Introduction” on page 628](#), we can see that most of the benefits derive from the single registry. Here is the list again, this time with items that are the direct result of a common registry *emphasized*:

- For end users
 - Only one authentication mechanism to remember. *For password-based authentication, this means users only have to remember one password.*
 - *If using passwords, users only have to update one password and follow one set of password rules.*
 - A single sign-on for each user into the SSO domain, typically only once per day.
- For IT operations
 - *A single common registry of user information (possibly replicated).*
 - *A single common way to manage user information.*
 - A single common security infrastructure.
- Security advantages

- A common secure infrastructure leveraged enterprise wide that can be carefully managed and secured.
- Depending on the security approach taken, secure delegation of credentials is possible. This will enable end-to-end security, possibly across application and system boundaries.
- *Easier to manage and protect common registry.*
- *Easier to verify user security information and update when necessary rather than tracking down all operational systems. This is particularly valuable when users move to new roles with different access levels.*
- *Can enforce common enterprise wide password and security policies.*
- *Users less likely to write down passwords because they only have to remember one.*

Interestingly, from a business perspective, the emphasized items are the ones with the greatest cost benefit. While a complete solution is most desirable, this comes remarkably close. It is especially worth noting that while end users get most of the benefits of SSO with a single registry, an organization can also significantly reduce the operational cost of registry maintenance simply by consolidating to single registry. For the end user, they must authenticate multiple times per day using the same user name and password, which, while irritating, is not a significant cost.

It is also worth noting that a proxy-SSO solution can be built on top of a common registry solution. This will greatly

simplify the proxy-SSO problem and result in a better overall infrastructure. However, it is not clear if this is worth the additional cost given the limited benefit.

Of course, there is no panacea. Using a common registry requires that all existing applications be migrated to use this new registry and some organization must manage this critical business system. While not trivial, this cost will be leveraged across numerous applications that no longer have to manage a user registry.

The best registry to choose is one that supports Lightweight Directory Access Protocol (LDAP). There are a number of LDAP server products that can be purchased. Additionally, most security products and middleware products that support secure access can use an LDAP directory for authentication. Once an enterprise embarks on this effort, they may find that many applications can switch easily to LDAP simply by reconfiguring the middleware that they use. For example, Netscape Enterprise Server, Apache, IBM HTTP Server, AIX, Solaris, and WebSphere Application Server already support LDAP out of the box. Additionally, Microsoft Active Directory can provide LDAP services to other clients.

How to select an LDAP directory product is beyond the scope of this redbook. As with any enterprise class product, you should examine vendor support, scalability, performance, replication, fault tolerance, extensibility, security, tools, and features. In particular, you will want to consider password management features, such as account lockout and strength rules. These are not part of the LDAP standard, so they will be vendor specific.

For applications that use custom developed security registries, change will be required. This fact alone should raise concerns. If an application team develops the registry,

how does the enterprise security team know that the registry is well designed and well managed?

Fortunately, LDAP application programming interfaces (APIs) are available for many programming languages, including C/C++, Java, and Perl. The APIs are quite trivial to use because of their simplicity, so conversion of existing applications is feasible. While it is not realistic to believe that all applications can be converted, it is reasonable to expect that future applications will use a common registry. Additionally, most PKI based systems require a registry for user information and LDAP is widely supported. Because many organizations are moving toward PKI anyway, this can be viewed as a stepping-stone toward a full PKI-based security infrastructure. Existing applications can be converted to use the LDAP registry as appropriate, based on cost and benefit. Over time, this will greatly improve the situation without introducing a large and complex system that will require ongoing maintenance without significantly improving the situation. For those considering an SSO solution, now is the time to act. With each passing day, more and more incompatible systems are being deployed in your enterprise.

Conclusion

The appendix has discussed two approaches to SSO: a homogenous infrastructure and proxy-SSO. The costs and benefits of these approaches have been discussed. A practical alternative to SSO involving a common registry has also been discussed.

We hope that you now more fully understand the issues, costs, and benefits surrounding SSO solutions. You have to look at your own organization and carefully consider the costs and benefits in order to derive a true and accurate picture. This will require careful analysis and experimentation by highly skilled people. Should you then decide to go forward with a proxy-SSO solution, you will at least understand the task that lies before you.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see [“How to get IBM Redbooks”](#) on [page 641](#).

- *All About Tivoli Management Agents*, SG24-5134
- *Centralized Risk Management using Tivoli Risk Manager 4.2*, SG24-0695
- *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*, SG24-4986s
- *IBM Tivoli Privacy Manager Solution Design and Best Practices*, SG24-6999
- *Enterprise Business Portals with IBM Tivoli Access Manager*, SG24-6556
- *Enterprise Business Portals II with IBM Tivoli Access Manager*, SG24-6885
- *Identity Management Design Guide with IBM Tivoli Identity Manager*, SG24-6996

- *IBM WebSphere V5.0 Security WebSphere Handbook Series*, SG24-6573
- *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*, SG24-6077

Other resources

These publications are also relevant as further information sources:

- Keys Botzum, Single Sign-On - A contrarian view, IBM whitepaper. For more resource information see <http://www.transarc.ibm.com/~keys>.
- Lloyd and Galambos, “*Technical Reference Architectures*,” *IBM Systems Journal* 38, No. 1, 51–75 (1999), G321-0134
- Eberhardt Rechtin, *Systems Architecting: Creating and Building Complex Systems*, Prentice Hall, 1991, ASIN 0138803455
- Fred B. Schneider, *Trust in Cyberspace*, National Academy Press, 1999, ISBN 0309065585

These publications are packaged with their corresponding software and cannot be purchased separately:

- *IBM Tivoli Access Manager Base Administration Guide Version 5.1*, SC32-1360

- *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 5.1*, SC32-1359
- *IBM Tivoli Access Manager for e-business Plug-in for Web Servers Integration Guide Version 5.1*, SC32-1365
- *IBM Tivoli Access Manager for e-business Administration C API Developer Reference Version 5.1*, SC32-1357
- *IBM Tivoli Access Manager Administration Java Classes Developer Reference Version 5.1*, SC32-1356
- *IBM Tivoli Access Manager for e-business BEA WebLogic Server Integration Guide Version 5.1*, SC32-1366
- *IBM Tivoli Access Manager for e-business IBM WebSphere Application Server Integration Guide Version 5.1*, SC32-1368
- *IBM Tivoli Access Manager for e-business Problem Determination Guide Version 5.1*, SC32-1352
- *IBM Tivoli Access Manager for e-business Performance Tuning Guide Version 5.1*, SC32-1351
- *IBM Tivoli Access Manager for e-business IBM Tivoli Identity Manager Provisioning Fast Start Guide Version 5.1*, SC32-1364
- *IBM Tivoli Risk Manager Administrator's Guide Version 4.2*, GC32-1323
- *IBM Tivoli Directory Server Installation and Configuration Guide Version 5.2*, SC32-1338

- *IBM Tivoli Directory Server Administration Guide Version 5.2*, SC32-1339
 - *IBM Tivoli Directory Integrator 5.2: Reference Guide*, SC32-1377
 - *IBM Tivoli Directory Integrator 5.2: Administrator Guide*, SC32-1379
 - *Tivoli Identity Manager Policy and Organization Administration Guide Version 4.5*, SC32-1149
 - *IBM Tivoli Identity Manager Tivoli Access Manager Agent for Windows Installation Guide Version 4.5*, SC32-1165
 - *Tivoli Event Integration Facility Reference Version 3.9*, SC32-1241
-

Referenced Web sites

These 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/>

- Anti-Virus.com home page

<http://Anti-Virus.com>

- CERT Coordination Center home page

<http://www.cert.org>

- Check Point home page

<http://www.checkpoint.com>

- Cisco Systems home page

<http://www.cisco.com>

- CNET.com home page

<http://www.cnet.com>

- IBM alphaWorks Web page

<http://www.alphaworks.ibm.com>

- IBM Security Software Web page

<http://www.ibm.com/software/security>

- IBM WebSphere Application Server home page

<http://www.ibm.com/software/webservers/appserv/>

- IBM WebSphere Edge Server home page

<http://www.ibm.com/software/webservers/edgeserver/>

- Lotus Domino R5 Application Server Web page

<http://www.lotus.com/home.nsf/welcome/dominoapplicationserver>

- 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>

- Panda Security home page

<http://www.pandasecurity.com/gviinfo.htm>

- RFC home page

<http://www.ietf.org/rfc.html>

- RSA Keon Web page

<http://www.rsasecurity.com/products/keon/index.html>

- Symantec home page
<http://www.symantec.com>
- Symantec products home page
<http://www.symantec.com/product/>
- Tivoli Risk Manager Web page
<http://www-306.ibm.com/software/tivoli/products/risk-mgr/detail.html>
- Trend Micro product home page
<http://www.antivirus.com/products/>
- VeriSign Managed PKI home page
<http://www.verisign.com/products/pki/index.html>
- WebSphere home page
<http://www-3.ibm.com/software/info1/websphere/index.jsp>
- WebSphere Application Server - Software Prerequisites Web page
http://www-3.ibm.com/software/webservers/appserv/doc/v35/idx_aas.htm
- WebSphere Application Server - Supported prerequisites and APIs Web page
http://www-3.ibm.com/software/webservers/appserv/doc/v40/prereqs/ae_v402.htm

- ZDNET.com home page

<http://www.zdnet.com>

How to get IBM Redbooks

Search for additional Redbooks or Redpieces, view, download, or order hardcopy from the Redbooks Web site:
ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become Redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all of the CD-ROMs offered, as well as updates and formats.

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 Lists (ACL)

A file attribute that contains the basic and extended permissions that control access to the file.

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.

Agent

A function that represents a requester to a server. An agent can be present in both a source and a target system.

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.

BS7799

British Standard 7799, a document describing enterprise security.

CA

An authority in a network that 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.

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

Common Object Request Broker Architecture is 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.

CTCPEC

Canadian Trusted Computer Product Evaluation Criteria published by the Canadian government.

DHCP (Dynamic Host Configuration Protocol)

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.

DMZ

A demilitarized zone is an area of your network that separates it from other areas of the network, including the Internet.

EJB

Enterprise JavaBeans is 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 offers 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.

ITSEC

Information Technology Security Evaluation Criteria, published by the European Commission.

J2EE

Java 2 Platform Enterprise Edition is 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.

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.

JDBC

Java Database Connectivity is 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.

JNDI

Java Naming and Directory Interface 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).

JSP

Java Server Page is 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.

LDAP

Lightweight Directory Access Protocol is 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.

LTPA

Lightweight Third Party Authentication 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.

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.

MASS

Method for Architecting Secure Solutions

NAT (Network Address Translation)

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.

NIS

Network Information System 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.

OPSEC

Open Platform for Security Check Point initiative to provide a common architecture for integrating security solutions.

OSI

Open Systems Interconnection is 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.

PKI

A public key infrastructure enables users of a basically unsecure public network such as the Internet, to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority.

RA

A Registration Authority is an authority in a network that verifies user requests for a digital certificate and tells the certificate authority (CA) to issue it. RAs are part of a public key infrastructure (PKI), a networked system that enables companies and users to exchange information and money safely and securely. The digital certificate contains a public key that is used to encrypt and decrypt messages and digital signatures.

RBAC

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.

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.

SOAP

Simple Object Access Protocol is 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.

SPI (Stateful Packet Inspection)

A firewall technology that examines the content of packets to determine whether they will be given access to a network.

SSL

The Secure Sockets Layer is 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.

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.

UDDI

Universal Description, Discovery, and Integration is 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.

WAP

Wireless Application Protocol is 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, newsgroups, 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.

WML

Wireless Markup Language, formerly called HDML (Handheld Devices Markup Languages), is a language that allows the text portions of Web pages to be presented on cellular telephones and personal digital assistants (PDAs) via wireless access. WML is part of the Wireless Application Protocol (WAP) that is being proposed by several vendors to standards bodies.

WSDL

The Web Services Description Language is 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.

XML

eXtensible Markup Language is 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.

XSL

eXtensible Stylesheet Language is a language for creating a style sheet that describes how data sent over the Web using the eXtensible Markup Language (XML) is to be presented to the user.

Index

A

- Abstract Syntax Notation One [90](#)
- access control [4](#), [19](#), [20](#), [23](#), [32](#), [37](#), [43](#), [57](#), [149](#), [216](#)
 - decision function [264](#)
 - decision information [264](#), [279](#)
 - enforcement function [264](#)
 - mechanisms [29](#)
 - monitoring [29](#)
 - ruleset [135](#)
 - use case [134](#)
- access control information [109](#)
 - filtered ACL [110](#)
 - non-filtered ACL [110](#)
- access control list
 - see [ACL](#)
- access control subsystem [134](#)
 - security design objectives [205](#)
- access evaluation [112](#)
- Access Manager [161](#)
 - ... for BEA WebLogic [158](#)
 - ... for Business Integration [367](#)
 - ... for Operating Systems [339](#)
 - ... for WebSphere Application Server [158](#), [304](#)
 - ... for WebSphere Business Integration Brokers [367](#), [386](#)
- ACL [141](#), [271](#)
- ACL database caching [176](#)
- ACL DB [265](#)
- ACL evaluation [275](#)
- ACL policy [275](#)
- administration [269](#)

administration API [138](#), [161](#), [267](#)
administrator levels [146](#)
any-other [276](#)
architecture [178](#), [316](#)
Attribute Retrieval Service [282](#)
audit daemon [351](#)
audit for MQ [383](#)
auditing [355](#)
audit-level policy [273](#)
authentication [235](#)
authentication for MQ [383](#)
authentication strength [277](#)
authorization API [267](#)
authorization components [267](#)
authorization daemon [350](#)
authorization database [141](#), [142](#)
authorization evaluator [266](#)
authorization flow [272](#)
authorization for MQ [383](#)
authorization rule [141](#), [271](#), [277](#)
authorization rule policy [273](#)
authorization server [173](#)
authorization server availability [224](#)
authorization service [137](#), [143](#), [149](#), [161](#), [265](#)
authorization service interface [267](#)
availability [215](#), [217](#)
aznAPI [161](#), [267](#), [303](#)
base components [136](#)
CDMF [329](#)
CDSSO [316](#), [327](#)
communication [179](#)
component deployment [177](#)
Cross Domain Single Sign-On [327](#)
cross-site fail-over [319](#)
data protection for MQ [383](#)
default domain [270](#)

design principles [132](#)
distributed communication [322](#)
domain [268](#)
e-Community single sign-on [316](#)
entitlement service [161](#)
entitlement service interface [250](#)
entitlements [279](#)
error handling for MQ [384](#)
extended attributes [141](#)
external authorization service [273](#)
fail-over [319](#)
file policy [345](#)
GSO junction [296](#)
hardening [197](#)
Identity Manager integration [488](#)
interceptor model for MQ [382](#)
interfaces [160](#)
intervention point [353](#)
Introduction [127](#), [263](#), [339](#), [393](#)
IP endpoint authentication method [272](#)
J2EE [294](#)
JAAS [267](#), [294](#), [295](#)
JMS Interceptor for MQ [385](#)
Kerberos authentication [249](#)
log router daemon [352](#), [358](#)
login policy [348](#)
login policy and password management daemon
[351](#)
LTPA [295](#), [301](#)
Management Domain [270](#)
Master Authentication Server [331](#)
MQ Client Interceptor [384](#)
multiple directories [511](#)
multiple domain support [268](#), [312](#), [327](#), [362](#)
network policy [347](#)
network-based authentication [277](#)

NTLM authentication [249](#)
password management policy [349](#)
pdadmin [137](#), [143](#)
PDPPermission [295](#), [304](#)
physical component layout [195](#)
pkmscdsso [328](#)
pkmsvouchfor [332](#)
Plug-In for Web servers [154](#), [173](#)
policy branches [362](#)
policy database [265](#)
Policy Server [137](#), [142](#), [170](#), [180](#), [265](#), [319](#)
Policy Server availability [228](#)
POP ... see [protected object policy](#)
privacy services [406](#)
protected object policy [161](#), [271](#), [276](#), [355](#)
protected object space [140](#), [148](#), [161](#), [229](#), [267](#), [273](#)
protected object space guidelines [283](#)
Proxy Policy Server [137](#), [143](#), [175](#), [180](#), [322](#)
quality of protection [277](#)
resource manager [127](#), [148](#)
scalability [215](#), [230](#)
security policy [178](#), [271](#)
Security Service Provider Interface [159](#)
single sign-on [168](#), [205](#), [296](#), [327](#)
SPNEGO protocol [248](#)
SSL [322](#)
step-up authentication [277](#)
su command [353](#)
sudo policy [349](#)
surrogate policy [349](#)
time-of-day policy [273](#)
Tivoli Enterprise Console daemon [351](#)
Trust Association Interceptor [294](#), [296](#)
unauthenticated [276](#)
user registry [170](#), [181](#)
virtual hosting [153](#)

VPN [323](#)
watchdog daemon [351](#)
Web Portal Manager [137](#), [144](#)
WebSEAL [148](#), [165](#)
Windows single sign-on [248](#)

Access Policy Evaluator [135](#)
access rights [111](#)
access targets [111](#)
accountability [62](#)
ACE/Server [248](#)
ACL [94](#), [141](#), [221](#), [229](#), [271](#), [345](#)
ACL database caching [176](#)
ACL DB [265](#)
Active Directory Changelog Connector [511](#)
administration [62](#)
administration API [138](#), [161](#), [267](#)
administration of a directory [102](#)
alarm handler [564](#)
AMPS [253](#)
analysis of security audit data [26](#)
analysis ruleset [564](#)
anonymity [32](#)
antivirus [530](#), [623](#)
application [90](#)
application components [399](#)
application layer firewall [53](#)
architectural decision [40](#)
architecture design principles [132](#), [341](#), [377](#)

ASN.1 [90](#)

asset classification [6](#)

asset protection [62](#)

assurance [59, 62](#)

audit [23, 25, 37, 43, 167, 169, 196, 216](#)

audit analyzer [564](#)

audit collector [563](#)

audit daemon [351](#)

audit log [135](#)

audit report generator [564, 569](#)

Audit System Interface [135](#)

auditing [355](#)

 UNIX administrative activity [356](#)

 UNIX authorization decisions [355](#)

 UNIX trace events [357](#)

authentication [19, 20, 29, 32, 77, 149, 162, 167, 205, 235, 240, 243, 246, 250](#)

 basic [246](#)

 Directory Server [106](#)

 flexibility [237](#)

 Kerberos [249](#)

 MPA [252](#)

 none [252](#)

 NTLM [249](#)

 step-up [237](#)

 strength [237](#)

 WebSEAL [150, 240](#)

authenticity [77](#)

authoritative identity information [502](#)

authorization [62, 167](#)

 Access Control Decision Function [264, 279](#)

Access Control Enforcement Function [264](#)
API [264](#), [267](#)
daemon [350](#)
database [140](#), [141](#), [142](#), [149](#), [161](#)
Directory Server [109](#)
evaluator [266](#)
flow [272](#)
initiator [264](#)
local cache mode [225](#)
mechanisms [29](#)
remote cache mode [224](#)
rule [141](#), [271](#)
service [137](#), [143](#), [149](#), [161](#), [265](#)
service interface [267](#)
target [264](#)
authorization server [173](#)
scalability [231](#)
availability [62](#), [92](#), [215](#), [217](#), [219](#), [223](#)
Access Manager authorization server [224](#)
Access Manager Policy Server [228](#)
Policy Server [228](#)
user registry [226](#)
Web Portal Manager [229](#)
Web server [223](#)
WebSEAL [220](#)
Axent Technologies Raptor Firewall [620](#)
aznAPI [161](#), [288](#), [295](#), [303](#), [304](#)
 entitlement service interface [250](#)
Java 2 security model [161](#)

Index

B

- base components [136](#)
 - authorization database [140](#)
 - core components [137](#)
 - management components [137](#)
- basic authentication [246](#)
- Binding Enabler [135](#)
- British Standard 7799 [3](#)
 - BS7799 [3](#)
- bulk load [508](#)
- business continuity [4](#)
- business process [432](#)
 - flow [32](#)
 - re-engineering [437](#)
- business requirements [205](#)
 - access control application integration [286](#)
 - access control subsystem [205](#)
 - Access Manager for Operating Systems [358](#)
 - CDSSO [312](#)
 - centralized directory [84](#)
 - distributed audit environment [579](#)
 - e-Community [312](#)
 - identity management solution [483](#)
 - Web security [167](#)
 - WebSEAL [237](#)

Index

C

- C language [91](#)
- CA [622](#)
- CDAS [162](#), [248](#), [249](#)
- CDMA [253](#)
- CDMF [163](#)
- CDSSO [328](#)
- certificate [44](#), [78](#), [247](#)
- Certificate Authority [622](#)
- certificates
 - Tivoli Directory Server [108](#)
- channel exit [369](#)
- Check Point FireWall-1 [621](#)
- circuit level firewall [52](#)
- Cisco Local Director [220](#)
- Cisco PIX Firewall [620](#)
- client/server application [399](#)
- client/server model [90](#)
- collection of security audit data [26](#)
- Common Criteria [15](#), [20](#), [23](#), [58](#), [502](#)
- Common Object Request Broker Architecture [643](#)
- Common Vulnerabilities and Exposures [555](#)
- communication [20](#)
- compliance [5](#)
- component access [20](#)

component architecture [46](#)
computer management [6](#)
consent [401](#)
container-based authorization [304](#)
content filtering [620](#)
controlled network [60](#), [480](#)
controlled zone [59](#)
CORBA [72](#), [76](#), [643](#), [644](#)
core components [137](#)
correlation
 state-based [541](#), [551](#)
Credential Validator [135](#)
credentials [57](#)
 secure [259](#)
credentials life cycle [32](#)
Cross Domain Authentication Service [162](#), [248](#), [249](#)
Cross Domain Mapping Framework [163](#)
Cross Domain Single Sign-On [328](#)
crypt [108](#)
cryptographic [32](#)
cryptographic support [20](#)
cryptography [28](#), [29](#), [31](#)
Crystal Reports [544](#), [568](#), [587](#), [593](#)
custom authentication [249](#)
custom realm [159](#)

Index

D

data confidentiality [19](#)
data integrity [19](#)
 Directory Server [108](#)
database [88](#)
 general-purpose [88](#)
declarative security [291](#)
delegation [144](#), [238](#), [240](#), [245](#), [253](#)
 WebSEAL [240](#)
demilitarized zone [51](#), [525](#)
deployment descriptor [291](#), [305](#), [306](#)
design objectives [35](#), [37](#), [57](#)
 mapping to security subsystems [36](#)
 Web security [168](#)
DHCP [196](#)
digital certificate [44](#), [78](#)
digital signature [77](#)
directory
 administration [102](#)
 and databases [88](#)
 and transactions [88](#)
 availability and scalability [100](#)
 distributed [91](#)
 distributed administration [103](#)
 firewall configuration [98](#)
 namespace [95](#)
 naming style [97](#)
 partitioned and replicated [92](#)

partitioning [101](#)
physical architecture [97](#)
referral [101](#)
replication [100](#)
schema [94](#), [113](#)
security [92](#), [106](#)
servers and clients [90](#)
technologies [79](#), [83](#)
telephone [87](#)

Directory Enabled Network [113](#)

Directory Information Tree [95](#)

Directory Integrator [118](#), [438](#)

- Active Directory Changelog Connector [511](#)
- administration [463](#)
- Administration and Monitor Console [447](#), [463](#)
- AssemblyLine [448](#), [492](#)
- base components [447](#)
- connectors [450](#)
- data flow topology [456](#)
- data flows [446](#)
- data sources [445](#)
- DSML EventHandler [509](#)
- EventHandlers [452](#)
- events [446](#)
- general benefits [443](#)
- high availability [460](#)
- hooks [453](#)
- Identity Manager synchronization [511](#)
- JavaScript [450](#), [453](#)
- LDAP changelog connector [511](#)
- LDAP Connector [512](#)
- logging [462](#)
- metadirectory [441](#)
- metadirectory scenario [505](#)
- multiple server environment [459](#)

parser [451](#)
parser connector [477](#)
password management [513](#)
password synchronization [453](#)
PerlScript [450](#), [453](#)
reconciliation [509](#)
scalability [460](#)
scripts [453](#)
VBScript [450](#), [453](#)
Web portal enablement [518](#)

Directory Server [104](#), [140](#)
access control information [109](#)
access evaluation [112](#)
access rights [111](#)
access targets [111](#)
administration [119](#)
Administration Group [120](#)
authentication [106](#)
authorization [109](#)
availability [114](#)
base components [105](#)
directory security [106](#)
forwarder [114](#)
gateway topology [117](#)
integrity [108](#)
logging [118](#)
master - forwarder - replica topology [115](#)
master - replica topology [114](#)
multiple masters [116](#)
password encryption [108](#)
password policy [109](#)
peer replication [114](#)
peer topology [116](#)
proxy authorization [112](#)
pseudo DN [110](#)

replica [114](#)
scalability [114](#)
schema [113](#)
subject [110](#)
Web Administration Tool [119](#)
distributed administration [103](#)
distributed correlation engine [542](#), [567](#)
distributed directory [91](#)
distributed security domains [327](#)
DMZ [51](#), [60](#), [63](#), [166](#), [177](#), [480](#), [525](#)
DNS [195](#)
Domain Name Service [195](#)
domains
 home [333](#)
Domino [75](#)
DSML EventHandler [509](#)
dynamic business entitlements [246](#)
dynamic packet filter firewall [53](#)

Index

E

EAS [161](#)

e-business on demand [401](#)

e-business patterns [62](#)

e-Community single sign-on [152](#), [331](#)

Edge Server [220](#)

EJB [69](#), [70](#), [219](#), [304](#), [644](#)

 role-based security [291](#)

EJBContext [294](#)

electronic commerce [34](#)

encryption [77](#)

 private key [77](#)

 public key [77](#)

enforcement mechanisms [29](#)

Enterprise Archive files [290](#)

Enterprise Java Beans [69](#), [219](#), [644](#)

entitlement service [161](#)

entitlement service interface [250](#)

environmental security [5](#)

EPAC [138](#), [244](#), [247](#)

Error Handler [135](#)

Event Integration Facility [566](#), [582](#)

event management reports [595](#)

Everyplace Wireless Gateway [253](#)

extended attributes [141](#)

Extended Privilege Attribute Certificate [138](#), [244](#)

eXtensible Markup Language [646](#)

eXtensible Stylesheet Language [646](#)

External Authorization Service [161](#)

external zone [59](#)

Index

F

- failure [218](#)
 - failure recovery [27](#)
 - fault tolerance [27](#)
 - federation of identities [420](#)
 - firewall [620](#)
 - appliance [51](#)
 - application layer [53](#)
 - circuit level [52](#)
 - dynamic packet filter [53](#)
 - filters [185](#), [189](#)
 - management reports [594](#)
 - packet filter [51](#)
 - risk management [546](#)
 - flow control [23](#), [30](#), [37](#), [43](#), [57](#)
 - forms-based authentication [247](#)
 - fraud [57](#)
 - FTP [197](#)
 - functional design [44](#)
 - functional requirements
 - centralized directory [85](#)
-

Index

G

gateway topology [117](#)

Global Sign-On [245](#), [254](#)

GSKit6 [151](#)

GSM [253](#)

GSO

lockbox [254](#)

GSO junction [296](#)

Index

H

hacker attack [168](#)

hardware security modules [28](#)

Health Insurance Portability Accountability Act [378](#)

HIPAA [378](#), [406](#)

home domain [333](#)

host intrusion [549](#)

host intrusion detection [572](#)

hosting service [186](#)

HTTP headers

 client identity [245](#)

 client IP addresses [245](#)

HTTP variables [259](#)

Index

I

IBM Tivoli Access Manager
see [Access Manager](#)

IBM Tivoli Directory Integrator
see [Directory Integrator](#)

IBM Tivoli Directory Server
see [Directory Server](#)

IBM Tivoli Identity Manager
see [Identity Manager](#)

IBM Tivoli Privacy Manager for e-business
see [Privacy Manager](#)

IBM WebSphere Application Server
see [WebSphere Application Server](#)

IBM WebSphere Edge Server
see [WebSphere Edge Server](#)

IBM WebSphere MQ
see [WebSphere MQ](#)

IBM WebSphere Portal
see [WebSphere Portal](#)

iDEN [253](#)

identification [20, 29, 32](#)

identity [137](#)

identity and credentials [23, 32, 37, 43, 44](#)

identity data management [502](#)

identity federation [420](#)

identity lifecycle management [420](#)

identity management

access control management [436](#)
access request approval [427](#)
accountability [426](#)
administration [429](#)
approval process [427](#)
audit [429](#), [436](#)
connector [424](#)
distributed administration [429](#)
implementation plan [433](#)
metadirectory [441](#)
orphan accounts [426](#)
password management [425](#), [436](#)
process automation [427](#)
repositories [424](#)
risk assessment [422](#)
scenarios [505](#)
security policy [436](#)
target systems [436](#)
user administration policy automation [431](#)
user management [436](#)
workflow [427](#)

Identity Manager [438](#)
accelerating deployments [507](#)
Access Manager integration [488](#)
account management module [471](#)
agent connectivity [475](#)
Application Layer [470](#)
asynchronous messaging [473](#)
authentication module [473](#)
authorization module [473](#)
bulk load [508](#)
connectivity [475](#)
data join management module [472](#)
data join module [475](#)
data services module [474](#)

desktop module [470](#)
DSML [475](#)
DSML EventHandler [509](#)
DSMLv2 JNDI connector [510](#)
dynamic role [472](#)
entity management module [472](#)
form design module [470](#)
form rendering module [470](#)
identity management module [471](#)
Java Message Service [473](#)
JMS [473](#)
join engine [475](#)
LDAP directory [475](#)
lifecycle management [466](#), [484](#)
logical component architecture [468](#)
mail module [473](#)
menu system module [469](#)
messaging module [473](#)
modification cycle [467](#)
network zones [480](#)
organization tree module [470](#)
orphan accounts [426](#)
password management [513](#)
password management module [472](#)
policy management module [471](#)
policy module [474](#)
provisioning [474](#)
provisioning cycle [467](#)
reconciliation [476](#), [509](#)
remote services module [474](#)
reporting module [472](#)
request for information [471](#)
role [472](#)
scheduling module [474](#)
search module [470](#)
service layer [472](#)

status change [474](#)
synchronization with Directory Integrator [511](#)
system configuration module [472](#)
termination cycle [467](#)
Web User Interface Layer [469](#)
workflow design [469](#)
workflow management module [471](#)
workflow module [474](#)
worklist management module [471](#)
X.509 certificate authentication [473](#)
identity mapping [138](#)
imask [109](#)
incident event [535](#)
initiator [264](#)
integrated identity management [420](#)
interfaces [160](#)
 aznAPI [161](#)
 Java API [161](#)
Internet DMZ [236](#)
intervention point [353](#)
intrusion detection [547](#), [566](#), [572](#), [620](#)
 reports [594](#)
 system [53](#), [529](#)
Intrusion Detection Exchange Format [555](#)
IPSec [620](#)
ISO 17799 [4](#)
IT security architecture [19](#)
ITSEC [644](#)

Index

J

J2EE [68](#), [305](#), [644](#)

Access Manager [294](#)

declarative security [291](#)

deployment descriptor [291](#), [306](#)

EJBContext [294](#)

programmatic security [293](#), [305](#)

J2EE 1.2 [290](#)

JAAS [161](#), [267](#), [294](#), [295](#), [303](#)

Java

application [91](#)

Java 2 security model [161](#)

Java Authentication and Authorization Services [161](#), [267](#),
[294](#)

Java Database Connectivity [644](#)

Java Message Service [368](#)

Java Naming and Directory Interface [644](#)

Java Server Pages [146](#), [219](#), [644](#)

JDBC [644](#)

JNDI [91](#), [644](#)

JSP [219](#), [644](#)

junction [148](#), [171](#), [184](#), [221](#), [223](#), [245](#)

mutually authenticated [299](#)

stateful junction [223](#)

Index

K

Kerberos
authentication [249](#)

Key Recovery Module [623](#)

KRM [623](#)

Index

L

LDAP [139](#), [635](#), [644](#)

API [90](#), [91](#)

availability [320](#)

changelog connector [511](#)

communication ports [107](#)

configuration [227](#)

connector [512](#)

protocol or directory? [89](#)

referral [101](#)

Software Development Kit [103](#)

LDAP authentication

Tivoli Directory Server [106](#)

ldapmodify [103](#)

ldapsearch [103](#)

liability [8](#)

lifecycle management [466](#), [484](#)

Lightweight Directory Access Protocol [635](#), [644](#)

Lightweight Third Party Authentication [645](#)

local cache mode [225](#)

log handler [563](#), [566](#)

log router daemon [352](#), [358](#)

logging [133](#), [341](#), [350](#), [378](#)

 Directory Server [118](#)

login policy and password management daemon [351](#)

logindenry [356](#)

loginpermit [356](#)

log-out function [247](#)

Lotus Domino [75](#), [140](#), [260](#)

Lotus Script [76](#)

LTPA [259](#), [295](#), [301](#), [645](#)

Index

M

management components [137](#)
Management Console [144](#)
Management Domain [270](#)
Management Framework [530](#)
MAS [331](#)
MASS [15](#), [58](#), [501](#)
 access control [23](#), [37](#), [43](#)
 Access Control Ruleset [135](#)
 access control subsystem [134](#)
 Access Policy Evaluator [135](#)
 architectural decision [40](#)
 audit [23](#), [25](#), [37](#), [43](#)
 audit log [135](#)
 Audit System Interface [135](#)
 Authentication Manager [135](#)
 Binding Enabler [135](#)
 component architecture [46](#)
 Credential Validator [135](#)
 Error Handler [135](#)
 flow control [23](#), [30](#), [37](#), [43](#)
 functional design [44](#)
 identity and credentials [23](#), [32](#), [37](#), [43](#), [44](#)
 identity data management [502](#)
 Resource Manager [134](#)
 solution integrity [23](#), [27](#), [37](#)
 solution model [40](#)
 State Manager [135](#)
 subsystems [24](#), [134](#)
 use case [41](#)

master - forwarder - replica topology [115](#)
master - replica topology [114](#)
Master Authentication Server (MAS) [331](#)
message channel agent [369](#)
message protection [377](#)
Message Queueing Interface [368](#)
meta directory [80](#)
metadirectory [441](#), [505](#)
Method for Architecting Secure Solutions
see [MASS](#)
Microsoft Active Directory [140](#)
monitor [401](#), [408](#)
MPA [252](#)
MQSeries [375](#)
multiple domain support [268](#)
multiple LDAP masters [116](#)
multiple security domains [312](#), [327](#)
Multiplexing Proxy Agent [252](#)

Index

N

naming style [97](#)

NAT [52](#)

Netscape iPlanet Directory [140](#)

network

 architecture [55](#)

 boundaries [51](#)

 configuration [173](#)

 framework [50](#)

 management [6](#)

 security [177](#)

 shadow [319](#)

 zones [60, 480](#)

Network Address Translation [52, 319](#)

network zone

 controlled [60, 480](#)

 restricted [60, 481](#)

 secured [61](#)

 uncontrolled [60, 480](#)

nonrepudiation [19](#)

Novell eDirectory [140](#)

NTP [196](#)

Index

O

- object authority manager [369](#)
 - OCSP [250](#)
 - ODBC [644](#)
 - Online Certificate Status Protocol [250](#)
 - Open Group
 - authorization API [264](#)
 - Open Systems Interconnection [645](#)
 - orphan accounts [426](#)
 - OS/390 Security Server [140](#)
 - OSI [645](#)
 - OSI security services [19](#)
-

Index

P

packet filter firewall [51](#)

PAM [246](#)

Panda Global Virus Insurance [625](#)

parser connector [477](#)

partitioning [92](#), [101](#)

password [256](#)

management [425](#), [513](#)

synchronization [453](#)

password encryption

 Directory Server [108](#)

password policy

 Directory Server [109](#)

pdadmin [137](#), [143](#)

PDC [253](#)

PDOSD daemon [344](#)

PDPermission [295](#), [304](#)

peer topology [116](#)

performance [92](#), [219](#)

perimeter network [51](#)

personnel security [5](#)

PHS [253](#)

physical architecture
 directory [97](#)

physical security [5](#)

PKI [44](#), [645](#)

PKIX [622](#)

Plug-in for Edge Server [153](#)

Plug-In for Web servers [154](#), [173](#)

plug-ins [186](#)

Plug-on Authentication Modules [246](#)

policy

branches [362](#)

corporate [8](#), [63](#)

database [265](#)

security [167](#)

Policy Server [137](#), [142](#), [170](#), [180](#), [265](#), [319](#)

availability [228](#)

failure [219](#)

POP [161](#), [229](#), [345](#)

see [protected object policy](#)

port restrictions for WebSEAL [211](#)

port scanning [54](#)

practices [9](#)

privacy [20](#), [57](#)

centralized service [401](#)

management components [403](#)

monitor [401](#)

Privacy Manager [394](#)

Access Manager services [406](#)

administrative users [404](#)

architecture [398](#)

audit [397](#)

auditor role [404](#)

centralized privacy service [400](#)

components [403](#)

consent [397](#)

console [403](#)

dedicated server configuration [407](#), [408](#)
Enterprise Java Bean container [409](#)
enterprise server configuration [412](#)
high availability [413](#)
HIPAA [406](#)
monitor [397](#), [402](#), [405](#)
monitor request [404](#)
monitor SDK [405](#), [408](#)
monitored system [403](#)
network protocols [410](#)
OECD principles [394](#)
P3P model [395](#)
P3P policies [405](#)
physical component architecture [407](#)
policy conformance check [406](#)
policy enforcement point [405](#)
privacy database [405](#)
Privacy Database Server [409](#)
privacy management architecture [401](#)
privacy policy [396](#), [409](#)
Privacy Server [403](#), [409](#)
purpose [404](#), [405](#)
Reference Monitor [406](#)
report definition [405](#)
reporting [397](#)
scalability [413](#)
storage location [404](#)
private key encryption [77](#)
procedures [9](#)
programmatic security [293](#), [305](#)
protected object policy [141](#), [161](#), [271](#), [276](#), [355](#)
protected object space [140](#), [148](#), [161](#), [221](#), [229](#), [267](#), [273](#)
guidelines [283](#)
protection of security audit data [26](#)

proxy authorization [112](#)
Proxy Policy Server [137](#), [143](#), [175](#), [180](#), [322](#)
proxy-SSO [629](#)
pseudo DN [110](#)
pseudonymity [32](#)
public key
 encryption [77](#)
 infrastructure [645](#)

Index

Q

query_contents [149](#)

quotas [28](#)

Index

R

RA [622](#), [645](#)
RACF [249](#)
RADIUS [249](#)
Raptor Firewall [620](#)
reconciliation [505](#), [509](#)
recovery [27](#)
Redbooks Web site [641](#)
 Contact us [xxi](#)
referral [92](#), [101](#)
Registration Authority [622](#), [645](#)
remote cache mode [224](#)
replication [92](#), [100](#)
Resource Access Control Facility [249](#)
resource manager [127](#), [134](#), [148](#), [288](#), [290](#)
resource utilization [20](#)
restricted network [60](#), [481](#)
restricted zone [59](#)
reverse proxy [148](#), [169](#), [186](#), [239](#)
risk assessment [422](#)
risk assessment reports [594](#)
risk management [59](#)
Risk Manager [569](#)
adapter [540](#), [566](#)
agent [539](#)

alarm handler [564](#)
analysis ruleset [564](#), [567](#)
architecture [538](#)
archive database [543](#)
audit analyzer [564](#)
audit collector [563](#), [566](#)
audit database [568](#)
audit report generator [564](#)
benefits [545](#)
class category [551](#)
client [541](#)
console [569](#)
Crystal Reports [544](#), [568](#), [587](#), [593](#)
CVE [555](#)
distributed correlation engine [542](#), [567](#)
Event Integration Facility [566](#), [582](#)
event monitor [543](#)
event processing [544](#)
event server [543](#), [551](#)
firewall management [546](#)
format file [541](#)
gateway [542](#)
historical reporting [584](#)
host intrusion [549](#)
host intrusion detection [572](#)
IDEF [555](#)
incident [537](#), [542](#), [551](#), [567](#), [583](#)

incident event [535](#)
incident group [553](#), [567](#)
intrusion detection [547](#), [566](#), [572](#)
log handler [563](#), [566](#)
reporting [543](#)
reports [588](#), [593](#)
risk assessment [550](#)
rm_Level attribute [551](#)
scalability [581](#)
sensor [566](#)
sliding-window [551](#)
summarization [566](#), [582](#)
summarization engine [541](#)
threshold [583](#)
Tivoli Data Warehouse [544](#), [584](#)
Tivoli Enterprise Console [535](#)
virus management [549](#)
Web server intrusion [548](#)
WebSEAL adapter [572](#)
Zurich Correlation Engine [540](#)
risk mitigation [6](#)
Role-Based Access Control [431](#), [645](#)
role-based security [291](#)
router [54](#)
RSA ACE/Server [248](#)
RSA Keon [622](#)
RSA SecurID token [248](#), [621](#)

Index

S

scalability [215](#), [230](#)
 authorization server [231](#)
 user registry [232](#)
 Web server [231](#)
 WebSEAL [230](#)

schema [94](#), [113](#)

secrets [32](#)

secure credential exchange [259](#)

secure credentials [259](#)

Secure Sockets Layer [151](#), [646](#)
 hardware acceleration support [151](#)
 Tivoli Directory Server [107](#)

secured network [61](#)

secured zone [59](#)

SecurID token [248](#)

security
 architecture [19](#), [34](#), [432](#)
 audit [20](#)
 audit data [26](#)
 cost [168](#)
 management [168](#)
 organization [6](#)
 policy [6](#), [61](#), [92](#), [167](#), [168](#), [178](#), [271](#), [422](#), [436](#), [481](#)

security design objectives [35](#), [37](#), [57](#), [205](#)
 Access Manager for Operating Systems [359](#)
 WebSEAL [237](#)

security domains

distributed [327](#)
multiple [312](#), [327](#)

security functions
management [20](#)
protection [20](#)

security subsystems mapping to design objectives [36](#)

sendmail [196](#)

server authentication
Tivoli Directory Server [106](#)

session cookie [245](#)

SHA-1 [108](#)

shadow network [319](#)

shadow standby network [319](#)

Simple Authentication and Security Layer
Tivoli Directory Server [108](#)

Simple Object Access Protocol [646](#)

single point of failure [217](#)

single sign-on [148](#), [150](#), [152](#), [168](#), [173](#), [205](#), [238](#), [245](#), [254](#),
[260](#), [296](#), [327](#), [627](#)
proxy-SSO [629](#)

SOAP [646](#)

solution architecture [40](#)

solution integrity [23](#), [27](#), [37](#)

solution model [40](#)

split-brain scenario [320](#)

SPNEGO protocol [248](#)

spoofing [53](#)

SQL [89](#)

SSL [151](#), [179](#), [322](#), [646](#)

hardware acceleration [151](#)
State Manager [135](#)
state-based
correlation [551](#)
state-based correlation [541](#)
stateful
inspection [53](#), [620](#)
junction [223](#)
packet filtering [169](#)
step-up authentication [237](#)
Structured Query Language (SQL) [89](#)
su command [353](#)
subject [110](#)
subsystems [24](#), [134](#)
Symantec Enterprise Firewall [620](#)
synchronization [505](#)
... of passwords [472](#)

Index

T

- TAR [248](#)
- target [264](#)
- TDMA [253](#)
- telephone directory [87](#)
- TGSO
 - password [256](#)
- time based log-out [247](#)
- time services [28](#), [196](#)
- time synchronization [196](#)
- Tivoli Adapter Configuration Facility [541](#)
- Tivoli Data Warehouse [544](#), [568](#), [583](#)
 - data collection [585](#)
 - data manipulation [586](#)
 - data reporting [586](#)
- Tivoli Enterprise Console [535](#)
 - daemon [351](#)
- Tivoli Global Sign-On [254](#)
- Tivoli Management Framework [530](#)
- token authenticator [248](#)
- transaction [88](#)
- Transaction Layer Security
 - Tivoli Directory Server [107](#)
- transactions [32](#)
- Trend Micro Antivirus [626](#)
- trust [242](#)

Trust Association Interceptor [294](#), [296](#)

Trusted Computing Base [346](#)

trusted credentials [57](#)

trusted path [20](#)

two-way authentication

Tivoli Directory Server [106](#)

Index

U

UDDI [646](#)

uncontrolled network [60](#), [480](#)

uncontrolled zone [58](#)

Universal Description, Discovery, and Integration [646](#)

use case [41](#)

 access control [134](#)

user administration policy automation [431](#)

user data protection [20](#)

user lifecycle management [419](#)

user registry [138](#), [170](#), [181](#)

 availability [226](#)

 failure [218](#)

 identity mapping [138](#)

 scalability [232](#)

 structure [139](#)

Index

V

- VeriSign OnSite [623](#)
 - virtual hosting [151](#), [153](#)
 - virus management [549](#)
 - virus management reports [597](#)
 - VPN [323](#), [620](#), [622](#)
-

Index

W

- WAP [253](#), [646](#)
- watchdog daemon [351](#)
- Web Administration Tool [119](#)
- Web Portal Manager [137](#), [144](#), [184](#), [219](#)
 - architecture [146](#)
 - availability [229](#)
 - Java Server Pages [146](#)
- Web security
 - business requirements [167](#)
 - design objectives [168](#)
 - principles [169](#)
- Web server [218](#), [223](#)
 - intrusion [548](#)
 - scalability [231](#)
- Web Server Plug-In architecture [154](#)
- Web services [68](#)
- Web Services Description Language [646](#)
- WebLogic
 - Access Manager for ... [158](#)
 - Security Service Provider Interface [159](#)
- WebSEAL [148](#), [165](#)
 - access control [149](#)
 - ACL [221](#)
 - adapter for Risk Manager [572](#)
 - adding a new WebSEAL [230](#)
 - authentication [149](#), [150](#), [162](#), [240](#), [243](#), [246](#), [250](#)
 - availability [220](#)
 - business requirements [237](#)

CDMF [329](#)
components for CDSSO and e-Community [316](#)
Cross Domain Single Sign-On (CDSSO) [328](#)
delegation [240](#), [245](#), [253](#)
e-Community Single Sign-On [331](#)
failure [218](#)
GSO junction [296](#)
hardening [197](#)
junction [148](#), [171](#), [184](#), [221](#), [223](#), [245](#)
LTPA [259](#), [295](#), [301](#)
Master Authentication Server [331](#)
mutually authenticated junction [299](#)
network configuration [173](#)
pkmscdsso [328](#)
pkmsvouchfor [332](#)
port restrictions [211](#)
query_contents [149](#)
replication [151](#)
reverse proxy [169](#), [236](#)
scalability [230](#)
security functions [149](#)
single sign-on [150](#), [260](#), [296](#)
SSL hardware acceleration [151](#)
trust [242](#)
Trust Association Interceptor [296](#)
webseald.conf [222](#), [230](#)

WebSphere
... Application Server [105](#), [119](#), [146](#)
... Edge Server [153](#), [220](#)
... Everyplace Suite [153](#), [253](#)
... family [66](#)
... MQ [368](#), [375](#)
... MQ Client [384](#)
... Portal [74](#)
... Service-Oriented Architecture [72](#)

... Studio Application Developer Integration Edition [73](#)
Access Manager for ... [158](#), [304](#)
Application Server [68-73](#)
Application Server - Enterprise [71-73](#)
Application Server - Express [70](#)
Application Server Network Deployment [71](#)
deployment descriptor [291](#)
Enterprise Archive files [290](#)
foundation and tools products [67](#)
 application development [67](#)
 enterprise modernization [68](#)
 open services infrastructure [67](#)
Trust Association Interceptor [296](#)
WebSphere Application Server
 configurations [69-73](#)
WebSphere Studio [73](#)
white pages [87](#)
Wireless Application Protocol [646](#)
Wireless Markup Language [646](#)
WML [646](#)
WSDL [646](#)

Index

X

X.500 [644](#)

X.500 directory [89](#)

X.509 [78, 247, 622](#)

XML [646](#)

XSL [646](#)

Index

Y

yellow pages [87](#)

Index

Z

z/OS Security Server [140](#)

Zurich Correlation Engine [540](#)

List of Figures

Chapter 1: Introduction

Figure 1-1: Risk categorization

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

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

Figure 1-4: Principal threat sources

Chapter 2: Method for Architecting Secure Solutions

[Figure 2-1:](#) IT security processes and subsystems

[Figure 2-2:](#) Security audit subsystem processes

[Figure 2-3:](#) Integrity subsystem processes

[Figure 2-4:](#) Access control subsystem processes

[Figure 2-5:](#) Information flow control subsystem processes

[Figure 2-6:](#) Credential subsystem processes

[Figure 2-7:](#) Networked information system environment

[Figure 2-8:](#) The normal and imperiled IT business process flow

[Figure 2-9:](#) Defending against attacks

[Figure 2-10:](#) Ensuring correct and reliable operation

[Figure 2-11:](#) Boundary flow control with security subsystems

[Figure 2-12:](#) Three-tier client/server input flow with security subsystems

[Figure 2-13:](#) Sample PKI digital certificate enrollment process flow

Chapter 3: IT Infrastructure Topologies and Components

[Figure 3-1:](#) Basic Internet boundary network configuration

[Figure 3-2:](#) High-level architectural model including all network components

[Figure 3-3:](#) How security fits into the enterprise

[Figure 3-4:](#) Applying MASS domain concepts Intranet

[Figure 3-5:](#) Graphic representation of network zones, transport classifications, and their level of trust

[Figure 3-6:](#) Basic DMZ design

[Figure 3-7:](#) Segregating the intranet client

[Figure 3-8:](#) Management zone, high availability, and load balancing

[Figure 3-9:](#) WebSphere family

[Figure 3-10:](#) WebSphere Application Server family

[Figure 3-11:](#) Horizontal and vertical portals

Chapter 4: Directory Technologies

Figure 4-1: Directory client/server interaction

Figure 4-2: Example of a Directory Information Tree (DIT)

Figure 4-3: Simple architecture with one LDAP server

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

Figure 4-5: Architecture with different internal and external users repository

Figure 4-6: Highly available and scalable LDAP server cluster

Figure 4-7: Master-replica topology

Figure 4-8: Cascading replication

Figure 4-9: Peer replication

Figure 4-10: Gateway replication M4

Chapter 5: Introduction to Access Manager Components

Figure 5-1: Access control subsystem

Figure 5-2: Relationship between the protected object space, ACLs, and POPs

Figure 5-3: Access Manager delegation model example

Figure 5-4: Web Portal Manager architecture

Figure 5-5: WebSEAL architecture

Figure 5-6: Plug-in for Edge Server architecture

Figure 5-7: Access Manager Web server plug-in architecture

Figure 5-8: WebSEAL EAS architecture

Figure 5-9: CDAS architecture

Chapter 6: Access Manager Web-based Architecture

[Figure 6-1:](#) Typical advanced Web application architecture

[Figure 6-2:](#) WebSEAL interaction with other Access Manager components

[Figure 6-3:](#) Direct serving of Web content from WebSEAL

[Figure 6-4:](#) Basic WebSEAL proxy functionality

[Figure 6-5:](#) Basic Web Server Plug-in components

[Figure 6-6:](#) Plug-in logical architecture

[Figure 6-7:](#) Communication flows using the Proxy Policy Server

[Figure 6-8:](#) Network zones

[Figure 6-9:](#) Policy Server placement guidelines

[Figure 6-10:](#) User Registry placement guidelines

[Figure 6-11:](#) Restricting network access to User Registry

[Figure 6-12:](#) Separating User Registry read and write functions

[Figure 6-13:](#) Web Portal Manager placement guidelines

[Figure 6-14:](#) Restricting HTTP/HTTPS network traffic paths

[Figure 6-15:](#) WebSEAL placement guidelines

[Figure 6-16:](#) Web server placement guidelines

[Figure 6-17:](#) Limiting network access to Web servers

[Figure 6-18:](#) Plug-in architecture

[Figure 6-19:](#) A sample Access Manager WebSEAL architecture

[Figure 6-20:](#) A sample physical component layout

[Figure 6-21:](#) Access Manager port matrix

Chapter 7: A Basic WebSEAL Scenario

[Figure 7-1:](#) Stocks-4u.com data network

[Figure 7-2:](#) Current Stocks-4u.com architecture

[Figure 7-3:](#) Initial WebSEAL architecture

[Figure 7-4:](#) WebSEAL security architecture with internal WebSEAL

[Figure 7-5:](#) Detailed WebSEAL security architecture with internal WebSEAL

[Figure 7-6:](#) WebSEAL physical architecture

Chapter 8: Increasing Availability and Scalability

[Figure 8-1:](#) Initial Web architecture

[Figure 8-2:](#) Server replication to increase availability

[Figure 8-3:](#) WebSEAL availability overview

[Figure 8-4:](#) WebSEAL availability configuration

[Figure 8-5:](#) Authorization Server scenario for Stocks-4U.com

[Figure 8-6:](#) Authorization Server scenario with high availability

[Figure 8-7:](#) Authorization Server scenario on local cache mode

[Figure 8-8:](#) Standby policy server configuration using a load balancer Other AM Components

Chapter 9: Authentication and Delegation with Access Manager

[Figure 9-1:](#) Reverse proxy flow for authentication, delegation, and authorization

[Figure 9-2:](#) Access Manager authentication methods with WebSEAL

[Figure 9-3:](#) Overview of Web server products protected with WebSEAL

[Figure 9-4:](#) Generic WebSEAL authentication model

[Figure 9-5:](#) WebSEAL authentication model with CDAS

[Figure 9-6:](#) Entitlement service

[Figure 9-7:](#) LDAP shared by Access Manager and other applications

[Figure 9-8:](#) Shared LDAP with separate user entries

[Figure 9-9:](#) WebSEAL LTPA token delegation

Chapter 10: Access Manager Authorization

[Figure 10-1:](#) Open Group authorization model

[Figure 10-2:](#) Access Manager authorization service

[Figure 10-3:](#) What is shared and separate per domain

[Figure 10-4:](#) Authorization decision flow

[Figure 10-5:](#) Access Manager protected object space

[Figure 10-6:](#) Authorization rules engine

Chapter 11: WebSphere Application Integration

[Figure 11-1:](#) Category 1 systems

[Figure 11-2:](#) Category 2 systems

[Figure 11-3:](#) Category 3 systems

[Figure 11-4:](#) Policy enforcement based on consistent decision making

[Figure 11-5:](#) Role-based security

[Figure 11-6:](#) TAI using a junction with the -b supply option

[Figure 11-7:](#) TAI using a junction without the -b supply option

[Figure 11-8:](#) TAI using a mutually authenticated SSL junction

[Figure 11-9:](#) WebSEAL creates LTPA cookies for authenticated users

[Figure 11-10:](#) Access Manager integration with WebSphere Application Server

[Figure 11-11:](#) Access Manager utility to map roles to principals and groups

[Figure 11-12:](#) Central administration for multiple WebSphere servers

Chapter 12: Access Control in a Distributed Environment

[Figure 12-1:](#) Stocks-4u.com security domain

[Figure 12-2:](#) Stocks-4u.com updated data network

[Figure 12-3:](#) IT center network zones

[Figure 12-4:](#) Distributed WebSEALs: replicated Web servers

[Figure 12-5:](#) Stocks-4u.com production network distributed server configuration

[Figure 12-6:](#) Cross-site SSL communication

[Figure 12-7:](#) Cross-site VPN communication

[Figure 12-8:](#) Bridged cross-site communication

[Figure 12-9:](#) Stocks-4u.com distributed WebSEAL architecture summary

[Figure 12-10:](#) CDSSO identity determination process

[Figure 12-11:](#) Stocks-4u.com CDSSO scenario

[Figure 12-12:](#) The e-community model

[Figure 12-13:](#) e-Community initial identity determination process

[Figure 12-14:](#) e-Community subsequent identity determination process

[Figure 12-15:](#) Stocks-4u.com e-community scenario

Chapter 13: Access Manager for Operating Systems

[Figure 13-1:](#) An overview of IBM Tivoli Access Manager for Operating Systems

[Figure 13-2:](#) Typical Access Manager for Operating Systems component interaction

[Figure 13-3:](#) Servers deployed within the Stocks-4u Web environment

[Figure 13-4:](#) Access Manager for Operating Systems domain structure

Chapter 14: Access Manager for Business Integration

[Figure 14-1:](#) IBM WebSphere MQ programming

[Figure 14-2:](#) WebSphere Business Integration Message Broker overview

[Figure 14-3:](#) Simple publish/subscribe system

[Figure 14-4:](#) IBM Tivoli Access Manager for Business Integration environment

[Figure 14-5:](#) Tivoli Access Manager for Business Integration interceptor model

[Figure 14-6:](#) IBM Tivoli Access Manager for Business Integration C Client Interceptor

[Figure 14-7:](#) IBM WebSphere MQ JMS architecture with JMS Client Interceptor

[Figure 14-8:](#) A Stocks-4u.com MQSeries application
Savannah

[Figure 14-9:](#) Access Manager for Business Integration architecture

[Figure 14-10:](#) A Stocks-4u.com Access Manager for Business Integration scenario

Chapter 15: Tivoli Privacy Manager

[Figure 15-1:](#) Solid security is the foundation of privacy protection

[Figure 15-2:](#) Basic components of a client-server application

[Figure 15-3:](#) Centralized privacy service

[Figure 15-4:](#) Layered architecture for privacy management

[Figure 15-5:](#) Components of a privacy management solution with IBM Tivoli products

[Figure 15-6:](#) Dedicated server configuration

[Figure 15-7:](#) Network Protocols between components of Dedicated Server Configuration

[Figure 15-8:](#) Enterprise Privacy Server configuration

Chapter 16: Identity Management

Figure 16-1: The eight levels of identity management

Figure 16-2: High-level and product-specific projects

Figure 16-3: Generic implementation phases for a project

Chapter 17: Introducing Directory Integrator

[Figure 17-1:](#) A general data integration environment

[Figure 17-2:](#) AssemblyLine

[Figure 17-3:](#) AssemblyLine with Connectors, Parsers, and EventHandlers

[Figure 17-4:](#) Password interception with Active Directory

[Figure 17-5:](#) Scenario with an enterprise directory

[Figure 17-6:](#) One-to-one integration

[Figure 17-7:](#) Many-to-one integration

[Figure 17-8:](#) One-to-many integration

[Figure 17-9:](#) Multiple server environment

[Figure 17-10:](#) High available servers

[Figure 17-11:](#) Scalable servers ITDI

Chapter 18: Identity Manager Structure and Components

[Figure 18-1:](#) User lifecycle management phases

[Figure 18-2:](#) Identity Manager logical architecture

[Figure 18-3:](#) Web User Interface module subprocesses

[Figure 18-4:](#) Applications Module subprocesses

[Figure 18-5:](#) Core Services module subprocesses

[Figure 18-6:](#) DSML connectivity to a service

[Figure 18-7:](#) IBM Directory Integrator service communication

Chapter 19: Identity Manager Scenarios

[Figure 19-1:](#) Network zones for Identity Manager placement

[Figure 19-2:](#) IBM Tivoli Identity Manager relationships

[Figure 19-3:](#) A workflow diagram example

[Figure 19-4:](#) Basic physical architecture for Areally Big Investment Security Corporation

[Figure 19-5:](#) Areally Big Investment Security Corporation basic architecture

[Figure 19-6:](#) Identity Manager as the central repository for user information

[Figure 19-7:](#) Network zones for Access Manager placement

[Figure 19-8:](#) Integrated architecture for Access Manager and Identity Manager

[Figure 19-9:](#) Sample Identity Manager and Access Manager integrated architecture

[Figure 19-10:](#) Basic high-level architecture compared side by side

[Figure 19-11:](#) Server component placement in each environment

[Figure 19-12:](#) Data synchronization focus

[Figure 19-13:](#) Directory Integrator data flows

Chapter 20: Synchronizing the Enterprise

[Figure 20-1:](#) MASS subsystem overview

[Figure 20-2:](#) Metadirectory services with Directory Integrator

[Figure 20-3:](#) Accelerating Identity Manager deployments

[Figure 20-4:](#) Multiple directories and Tivoli Access Manager

[Figure 20-5:](#) Password synchronization services

[Figure 20-6:](#) E-mail migration services

[Figure 20-7:](#) Enabling Web portals

Chapter 21: Risk Manager Topology and Infrastructure

[Figure 21-1:](#) e-business network topology

[Figure 21-2:](#) Tivoli management architecture

[Figure 21-3:](#) Tivoli Management Desktop

[Figure 21-4:](#) Enterprise console: all Risk Manager events

[Figure 21-5:](#) Risk Manager incident details

[Figure 21-6:](#) Risk Manager Architecture

[Figure 21-7:](#) How many incidents? How many from outside? How many from inside?

[Figure 21-8:](#) Internet connection increasingly cited as a frequent point of attack

[Figure 21-9:](#) Likely sources of attack

Chapter 22: Building a Centralized Security Audit Subsystem

[Figure 22-1:](#) Initial IT architecture for Stocks-4U.com.

[Figure 22-2:](#) Audit flow structure

[Figure 22-3:](#) Audit subsystem

[Figure 22-4:](#) Audit functions

[Figure 22-5:](#) Logic view with Risk Manager components

[Figure 22-6:](#) Risk Manager flow

[Figure 22-7:](#) Local summarization engine

[Figure 22-8:](#) Organization flows

Chapter 23: Extending the Centralized Security Audit Subsystem

[Figure 23-1:](#) Network integration architecture

[Figure 23-2:](#) Enhanced audit flow structure

[Figure 23-3:](#) Risk Manager and Tivoli Enterprise Console architecture

[Figure 23-4:](#) Tivoli Data Warehouse basic components

[Figure 23-5:](#) Tivoli Data Warehouse architecture

[Figure 23-6:](#) Policy / Configuration events, Last 30 Days

[Figure 23-7:](#) Events by Destination Host, Last 30 Days

[Figure 23-8:](#) Infection Events, Last 30 Days

[Figure 23-9:](#) Events by Destination and Category, Last 30 Days

[Figure 23-10:](#) Service Compromise Events, Last 30 Days

[Figure 23-11:](#) Top Web Intrusion Events

[Figure 23-12:](#) Top Web Intrusion Events, text format

[Figure 23-13:](#) Intrusion Events by Network Address

[Figure 23-14:](#) Top Firewall Events

[Figure 23-15:](#) Top events

[Figure 23-16:](#) Top Events text report

[Figure 23-17:](#) Events per Day by Address and Sensor Type

Figure 23-18: Organization flow

Chapter 24: Global MASS: An Example

Figure 24-1: Business view

Figure 24-2: Logical view

Figure 24-3: Detailed view

Figure 24-4: The full architectural view

List of Tables

Chapter 2: Method for Architecting Secure Solutions

Table 2-1: Functional category Common Criteria functional class

Table 2-2: Mapping design objectives to security subsystems

Table 2-3: Determining the security subsystems in a design

Chapter 4: Directory Technologies

Table 4-1: Example ACL for an employee's directory entry

Chapter 5: Introduction to Access Manager Components

Table 5-1: Delegated administration roles in Access Manager

Table 5-2: Plug-In for Web server functionalities

Chapter 11: WebSphere Application Integration

Table 11-1: LDAP directories supported by WebSphere and Access Manager

Chapter 13: Access Manager for Operating Systems

Table 13-1: File system permissions

Table 13-2: Network resource naming

Chapter 17: Introducing Directory Integrator

Table 17-1: Main available connectors

Chapter 20: Synchronizing the Enterprise

Table 20-1: Identity and credential mappings

List of Examples

Chapter 11: WebSphere Application Integration

Example 11-1: Method permissions in the ejb-jar.xml deployment descriptor

Example 11-2: Security constraints and permissions in a WAR deployment descriptor
