Lotus. software

IBM

Patterns: Custom Designs for Domino and WebSphere Integration

Harnessing the power of reusable assets

Selecting the best Patterns and products for your environment

Applying the Patterns to real-world scenarios

Tommi Tulisalo
Edward Cawthorne
Jonathan Czernel
Bradley Hertenstein
Kenneth Reed

Redbooks

ibm.com/redbooks

International Technical Support Organization

Patterns: Custom Designs for Domino and WebSphere Integration

April 2003

SG24-6903-00

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

First Edition (April 2003)

This edition applies to IBM Lotus Domino Server 6.0.1 and IBM WebSphere Application Server V5.0.

Contents

Notices

Trademarks

Preface

The Patterns for e-business are a group of proven, reusable assets that can speed the process of developing applications.

In this IBM® Redbook we describe the Application Integration patterns, and how, together with one or more of the other Patterns for e-business, they form Custom designs. We first discuss the application integration methods, how IBM Lotus® Domino™ 6 and IBM WebSphere® Application Server V5 can be integrated, and then move into Hybrid Runtime patterns, where both Domino and WebSphere Application Server exist. We then expand our discussion of Hybrid Runtime patterns to include Directory integration as well as Collaboration patterns.

In the second half of the book we describe three real-life scenarios where Patterns for e-business are applied and Domino and WebSphere Application Server are part of the runtime topology. We start from the business requirements, then identify and apply the applicable Business, Application, and Runtime patterns to get to our own runtime topology. We start with a simple scenario, and then increase the complexity in the following scenarios. We discuss technology options, and provide design and development guidelines for the solutions as well.

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, Cambridge Center.

Tommi Tulisalo is a project leader for the International Technical Support Organization at Cambridge, Massachusetts. He manages projects whose objective is to produce Redbooks on all areas of Lotus Software products. Before joining the ITSO in 2001, Tommi was an IT Architect for IBM Global Services in Finland, where he designed solutions for customers, often based on Lotus software.

Edward Cawthorne is an IT Specialist for Lotus Software working in Staines, United Kingdom. He has been with IBM since 1999, providing technical sales support for the entire Lotus Portfolio. Prior to joining IBM, Edward was with Derwent Information, where he worked on a project to deliver patent data in a Domino application. Previously, Edward served in the Royal Air Force.

Jonathan Czernel is a Senior IT Specialist for IBM Software Services for Lotus, with an emphasis on application architecture and development for Web browser and Lotus Notes®-based applications, as well as integration of Lotus Domino with back-end systems. He has served in various software engineering and technical leadership roles since 1987, and has extensive experience in several object-oriented and procedural development languages. He received his B.S. in Computer Science from the University of Missouri at Rolla, and is a Principal Certified Lotus Professional in Application Development (R5). Jonathan previously co-authored Lotus Notes 4.0 Unleashed (1996) and Microsoft ActiveX Programming Unleashed (1997), both published by Sams Publishing. Jon welcomes your feedback at jon_czernel@us.ibm.com.

Bradley Hertenstein is an I/T Architect for the AMS Architecture Center of Excellence within IBM Global Services. Bradley has over thirteen years of IT consulting experience in the areas of application development, systems administration, collaborative technologies, enterprise architecture integration, IT management, and emerging technologies. He currently specializes in e-business planning and implementation, and architectural design, applying IBM's integration methods based on Internet technologies, Domino, WebSphere, WebSphere Commerce Suite, Java and other emerging technologies. He is a Principal Certified Lotus Professional in System Administration R3-5, Principal Certified Lotus Professional in Application Development R3-5 and Certified Lotus Instructor R3-5. He will receive his B.S. in Economics from the University of South Florida in 2003.

Kenneth Reed is a Consulting I/T Specialist with IBM Software Services for Lotus (ISSL). Kenneth has over 25 years of experience in the architecture, design, development, and testing of software, primarily in the area of distributed objects in the context of frameworks. Among Kenneth's projects are an architecture for an enterprise-wide knowledge management framework, graphical design tools for database and object-oriented system designers, and a framework for the development of large-scale transaction processing systems. Kenneth is currently focusing on incorporating collaborative services into applications being developed for the WebSphere Application Server and WebSphere Portal. He received his B.S. in Mathematics from Northeastern University in Boston.

A number of people have provided support and guidance. In particular we would like to thank **Jonathan Adams**, IT consultant in IBM Software Group, United Kingdom and the leader of the Patterns for e-business initiative, for all the guidance and the careful review of the redbook.

Thanks also to the authors of the IBM redbook we have been updating. The redbook, *Applying the Patterns for e-Business for Domino and WebSphere Scenarios*, SG24-6255-00, was written by Søren Peter Nielsen, John Hartley, Andrew Larmour, Shankara Sudarsanam, Adrian Walmsley and Hisayoshi Yamamoto.

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

Alan Lepofsky, Gary Devendorf, Greg Kelleher, Roy Bowen, Michael Confoy, Vince Fertig - Lotus Software

Michele Galic - ITSO Raleigh

Alison Chandler - ITSO Poughkeepsie

Become a published author

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

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

1

Patterns for e-business

This redbook is part of the “Patterns for e-business” series. In this chapter we provide an introduction to the patterns and an overview of how IT architects can work effectively with them.

The role of the IT architect is to evaluate business problems and build solutions to solve them. To do this, the architect begins by gathering input on the problem, an outline of the desired solution, and any special considerations or requirements that need to be factored into that solution. The architect then takes this input and designs the solution. This solution can include one or more computer applications that address the business problems by supplying the necessary business functions.

To enable the architect to do this better each time, it is helpful to capture and reuse the experience of IT architects in such a way that future design efforts can be made simpler and faster. We do this by taking their experiences and using this knowledge to build a repository of assets. The repository provides a source where architects can learn from the experience of their peers to build future solutions, using proven assets. This reuse saves time, money, and effort, and in the process helps ensure delivery of a solid, properly architected solution.

The IBM Patterns for e-business help facilitate this reuse of assets. Their purpose is to capture and publish e-business artifacts that have been used, tested, and proven. The information captured by the patterns fits the majority of situations, and is further augmented with guidelines and related links.

Together, these tools allow the architect to start with a problem and a vision for the solution, and then find a pattern that fits that vision. Then, by drilling down using the patterns process, the architect can further define the additional functional pieces that the application will need to succeed. Finally, the architect can build the application using coding techniques identified in the associated guidelines.

1.1 The Patterns for e-business layered asset model

The Patterns for e-business approach enables architects to implement successful e-business solutions through the re-use of components and solution elements from proven successful experiences. The patterns approach is based on a set of *layered assets* that can be exploited by any existing development methodology. These layered assets are structured in a way that each level of detail builds on the last. These assets include:

- *Business patterns* identify the interaction between users, businesses, and data.
- *Integration patterns* tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern.
- *Composite patterns* represent commonly occurring combinations of Business patterns and Integration patterns.
- *Application patterns* provide a conceptual layout describing how the application components and data within a Business pattern or Integration pattern interact.
- *Runtime patterns* define the logical middleware structure supporting an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.
- *Product mappings* identify proven and tested software implementations for each Runtime pattern.
- *Best-practice guidelines* for design, development, deployment, and management of e-business applications.

These assets and their relation to each other are shown in Figure 1-1.

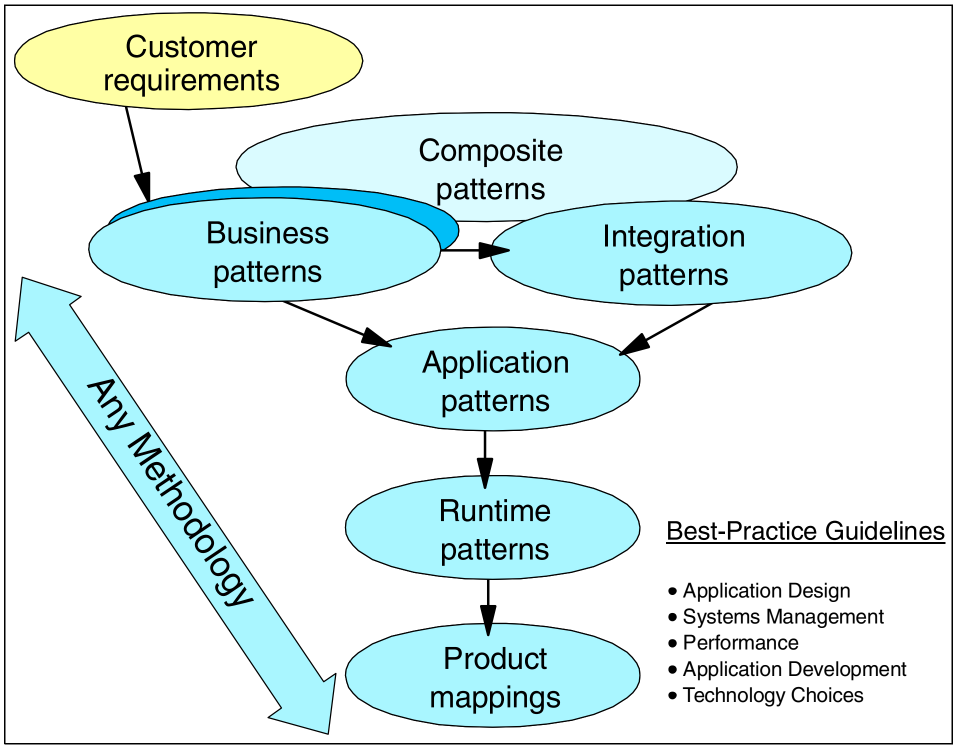


Figure 1-1 The Patterns for e-business layered asset model

Patterns for e-business Web site

The Patterns Web site provides an easy way of navigating top down through the layered Patterns' assets to determine the appropriate assets for a particular engagement.

For easy reference to Patterns for e-business, see the Web site at:

`http://www.ibm.com/developerWorks/patterns/`

1.2 How to use the Patterns for e-business

As described previously, the Patterns for e-business have a layered structure where each layer builds detail on the last. At the highest layer are Business patterns. These describe the entities involved in the e-business solution.

Composite patterns appear above Business patterns in the hierarchy diagram in Figure 1-1. However, Composite patterns are made up of a number of individual Business patterns, and at least one Integration pattern. In this section we discuss how to use the layered structure of the Patterns for e-business assets.

1.2.1 Select a Business, Integration, or Composite pattern, or a Custom design

When faced with the challenge of designing a solution for a business problem, the first step is to take a high-level view of the goals you are trying to achieve. A proposed business scenario should be described and each element should be matched to an appropriate IBM pattern for e-business. You may find, for example, that the total solution requires multiple Business and Integration patterns, or that it fits into a Composite pattern or Custom design.

For example, suppose an insurance company wants to reduce the amount of time and money spent on call centers that handle customer inquiries. By allowing customers to view their policy information and request changes online, they will be able to cut back significantly on the resources spent handling this by phone. The objective is to allow policyholders to view their policy information stored in legacy databases.

The "Self-Service business pattern" fits this scenario perfectly. It is meant to be used in situations where users need direct access to business applications and data. Let's take a look at the available Business patterns.

Business patterns

A Business pattern describes the relationship between the users, the business organization or applications, and the data to be accessed.

There are four primary Business patterns, identified in Table 1-1.

Table 1-1 Primary Business patterns

Business pattern	Description	Example
Self-Service (User-to-Business	Users interact with a business via the Internet	Simple Web site applications
Collaboration (User-to-User)	The Internet supports collaborative work between users	e-mail, community, chat, video conferencing

Business pattern	Description	Example
Information Aggregation (User-to-Data)	Users can extract useful information from large volumes of data, text, images and so forth	Business intelligence, knowledge management, Web crawlers
Extended Enterprise (Business-to-Business)	Applications that link two or more business processes across separate enterprises	EDI, supply chain management

It would be very convenient if all problems fit nicely into these four slots, but reality says that things will often be more complicated. The patterns assume that most problems, when broken down into their most basic components, will fit more than one of these patterns. When a problem requires multiple Business patterns, the Patterns for e-business provide additional frameworks in the form of Integration patterns.

Integration patterns

Integration patterns allow you to tie together multiple Business patterns to solve a business problem. The Integration patterns include those identified in Table 1-2.

Table 1-2 Integration patterns

Integration pattern	Description	Example
Access Integration	Integration of a number of services through a common entry point	Portals
Application Integration	Integration of multiple applications and data sources without the user directly invoking them	Message brokers, workflow managers

These Business and Integration patterns can be combined to implement installation-specific business solutions. We call this a *Custom design*.

Custom design

Figure 1-2 illustrates the use of a Custom design to address a business problem.

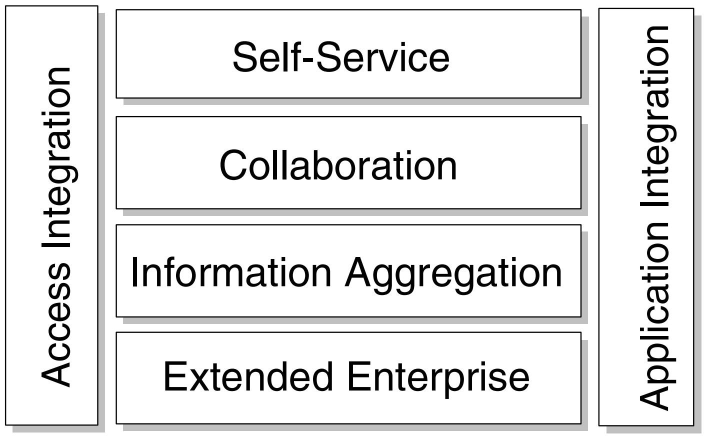


Figure 1-2 Patterns representing a Custom design

If any of the Business or Integration patterns shown in the diagram are not used in a particular Custom design, we show those blocks lighter than the others. For example, Figure 1-3 shows a Custom design that does not use a Collaboration business pattern or an Extended Enterprise business pattern for a business problem.

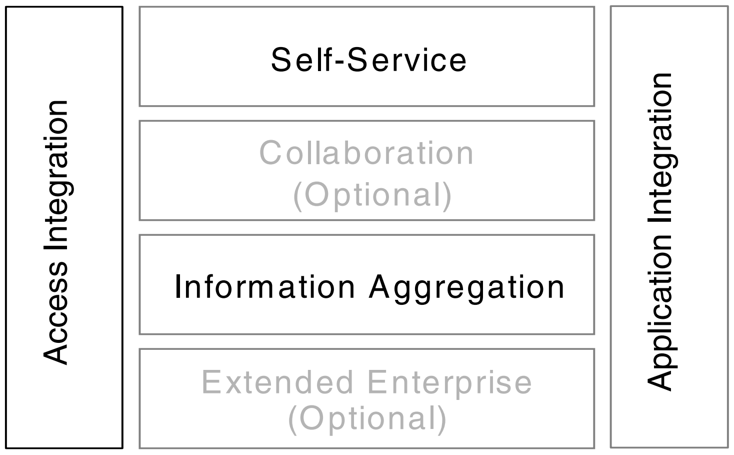


Figure 1-3 A Custom design using four of the Business and Integration patterns

A Custom design may also be a Composite pattern if it recurs many times across domains with similar business problems. For example, the iconic view of a Custom design in Figure 1-3 can also describe a Sell-Side Hub composite pattern.

Composite patterns

Several common uses of Business and Integration patterns have been identified and formalized into *Composite patterns*. The identified Composite patterns are described in Table 1-3.

Table 1-3 Composite patterns

Composite pattern	Description	Examples
Electronic Commerce	User to online buying	www.macys.com www.amazon.com
Portal	Typically designed to aggregate multiple information sources and applications to provide uniform, seamless, and personalized access for users	- Enterprise intranet portal providing Self-Service functions such as payroll, benefits, travel expenses - Collaboration providers who provide services such as e-mail or instant messaging
Account Access	Provide customers with around-the-clock access to their account information	- Online brokerage trading applications - Telephone company account manager functions - Bank, credit card, and insurance company online applications
Trading Exchange	Allows buyers and sellers to trade goods and services on a public site	- Buyer's side - interaction between buyer's procurement system and commerce functions of e-marketplace - Seller's side - interaction between procurement functions of the e-marketplace and its suppliers
Sell-Side Hub (Supplier)	The seller owns the e-marketplace and uses it as a vehicle to sell goods and services on the Web	www.carmax.com (a car purchase Web site)
Buy-Side Hub (Purchaser)	The buyer of the goods owns the e-marketplace and uses it as a vehicle to solicit the best deals for goods and services from prospective sellers across the Web	www.wre.org (WorldWide Retail Exchange)

The makeup of these patterns is variable in that there will be basic patterns present for each type, but the composite can easily be extended to meet additional criteria. For more information on Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos.

1.2.2 Selecting Application patterns

Once the Business pattern is identified, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the *Application pattern*. A Business pattern will usually have multiple possible Application patterns. An Application pattern may have logical components that describe a presentation tier for interacting with users, an application tier, and a back-end application tier.

Application patterns break the application down into the most basic conceptual components, identifying the goal of the application. In our example, the application falls into the Self-Service business pattern and the goal is to build a simple application that allows users to access back-end information. The Application pattern in Figure 1-4 fulfills this requirement.

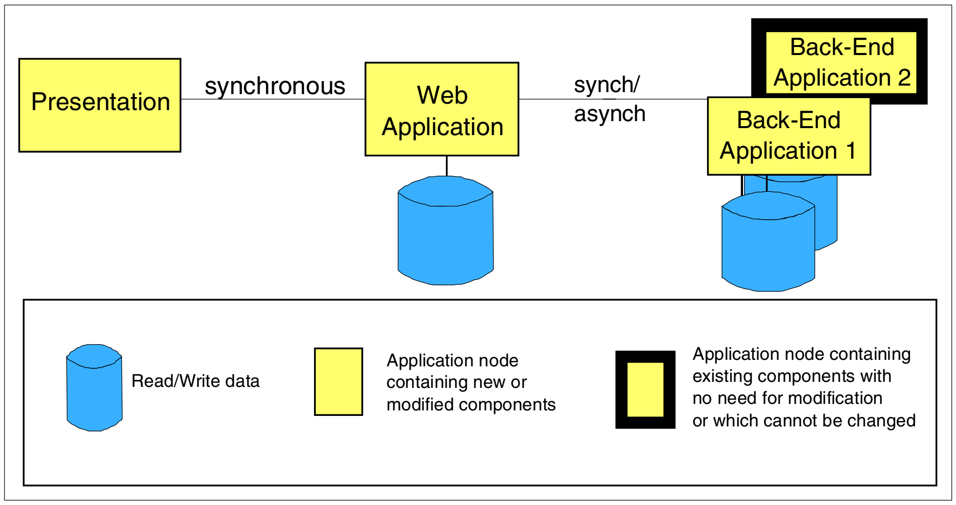

Presentation
synchronous
Web Application
synch/ asynch
Back-End Application 2
Back-End Application 1
Read/Write data
Application node containing new or modified components
Application node containing existing components with no need for modification or which cannot be changed

Figure 1-4 Self-Service::Directly Integrated Single Channel

The Application pattern shown includes a presentation tier that handles the request/response to the user. The application tier represents the component that handles access to the back-end applications and data. The multiple application boxes on the right represent the back-end applications that contain the business data. The type of communication is specified as synchronous (one request/one response, then next request/response) or asynchronous (multiple requests and responses intermixed).

Suppose that the situation is a little more complicated than that. Let's say that the automobile policies and the homeowner policies are kept in two separate and dissimilar databases. The user request would actually need data from multiple, disparate back-end systems. In this case there is a need to break the request down into multiple requests (decompose the request) to be sent to the two different back-end databases, then to gather the information sent back from the requests, and then put this information into the form of a response (recompose). In this case the Application pattern shown in Figure 1-5 would be more appropriate.

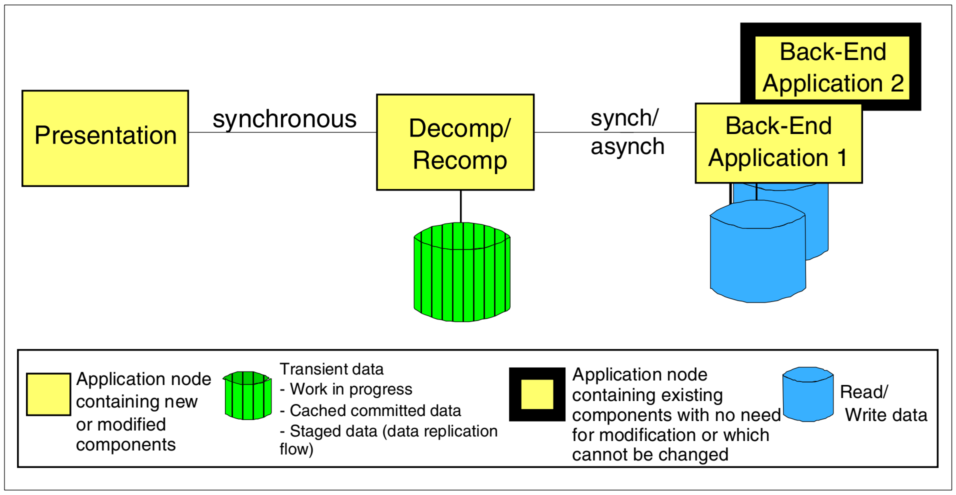

Presentation
synchronous
Decomp/ Recomp
synch/ asynch
Back-End Application 2
Back-End Application 1
Application node containing new or modified components
Transient data
- Work in progress
- Cached committed data
- Staged data (data replication flow)
Application node containing existing components with no need for modification or which cannot be changed
Read/ Write data

Figure 1-5 Self-Service::Decomposition

This Application pattern extends the idea of the application tier that accesses the back-end data by adding decomposition and recomposition capabilities.

1.2.3 Runtime patterns

The Application pattern can be further refined with more explicit functions to be performed. Each function is associated with a *runtime node*. In reality, these functions, or nodes, can exist on separate physical machines or they can co-exist on the same machine. In the Runtime pattern this is not relevant. The focus is on the *logical* nodes required and their placement in the overall network structure.

As an example, let's assume that our customer has determined that his solution fits into the Self-Service business pattern and that the Directly Integrated Single Channel pattern is the most descriptive of the situation. The next step is to determine the Runtime pattern that is most appropriate for his situation.

He knows that he will have users on the Internet accessing his business data, and he will therefore require a measure of security. Security can be implemented at various layers of the application, but the first line of defense is almost always one or more firewalls that define who and what can cross the physical network boundaries into his company network.

He also needs to determine the functional nodes required to implement the application and security measures. The Runtime pattern in Figure 1-6 is one of his options.

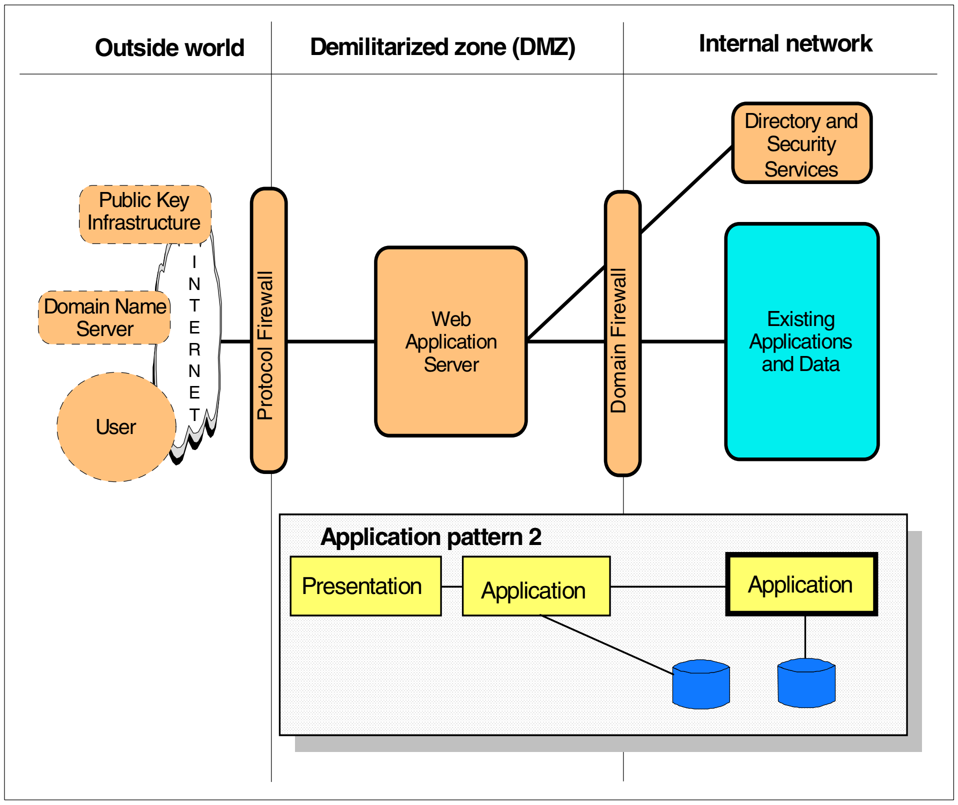


Figure 1-6 Directly Integrated Single Channel application pattern::Runtime pattern

By overlaying the Application pattern on the Runtime pattern, you can see the roles that each functional node will fulfill in the application. The presentation and application tiers will be implemented with a Web application server, which combines the functions of an HTTP server and an application server. It handles both static and dynamic Web pages.

Application security is handled by the Web application server through the use of a common central directory and security services node.

A characteristic that makes this Runtime pattern different from others is the placement of the Web application server between the two firewalls. The Runtime pattern shown in Figure 1-7 is a variation on this. It splits the Web application server into two functional nodes by separating the HTTP server function from the application server. The HTTP server (Web server redirector) will serve static Web pages and redirect other requests to the application server. It moves the application server function behind the second firewall, adding further security.

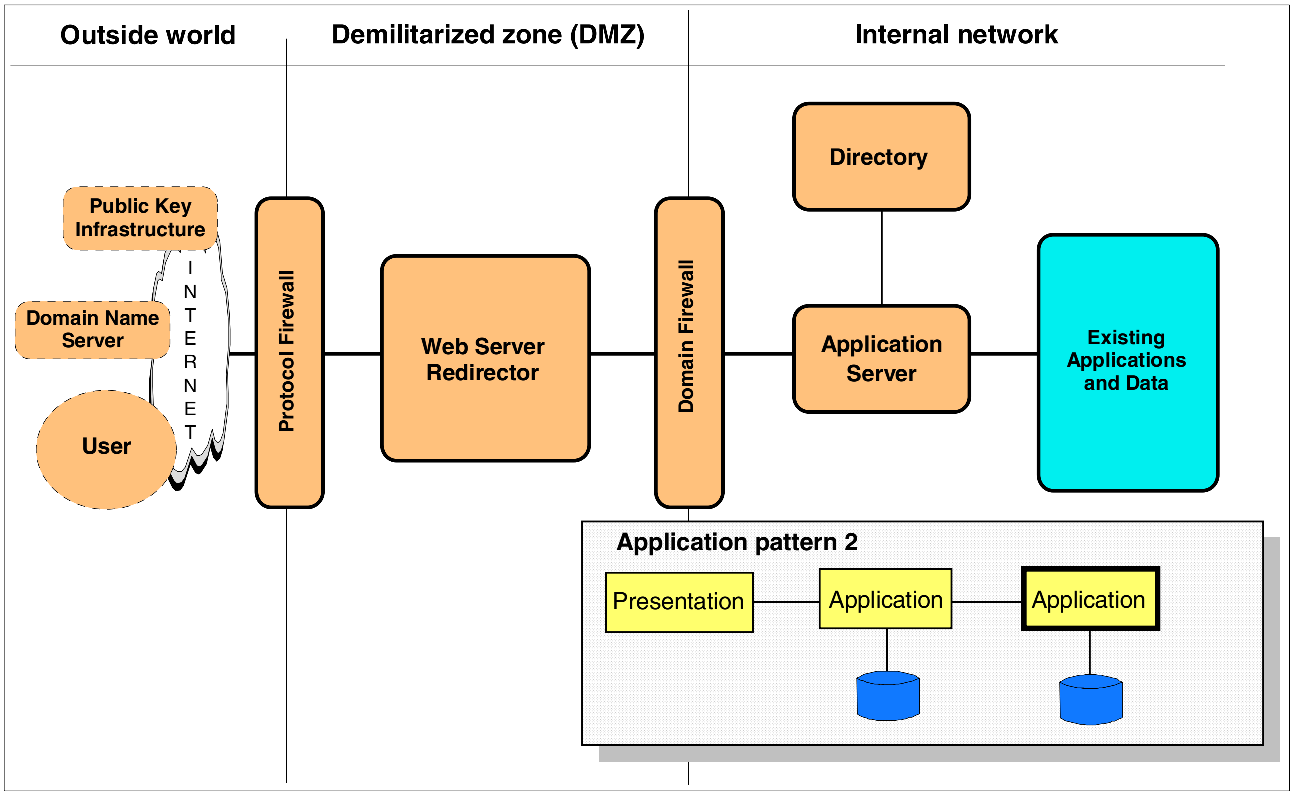


Figure 1-7 Directly Integrated Single Channel application pattern::Runtime variation 2

These are just two examples of the possible Runtime patterns available. Each Application pattern will have one or more Runtime patterns defined. These can be modified to suit the customer's needs. For example, the customer may want to add a load-balancing function and multiple application servers.

1.2.4 Product mappings

The last step in defining the network structure for the application is to correlate real products with one or more runtime nodes. The Patterns Web site shows each Runtime pattern with products that have been tested in that capacity. The product mappings are oriented toward a particular platform, though more likely, the customer will have a variety of platforms involved in the network. In this case, it is simply a matter of mixing and matching products.

For example, the Runtime variation in Figure 1-7 could be implemented using the product set depicted in Figure 1-8.

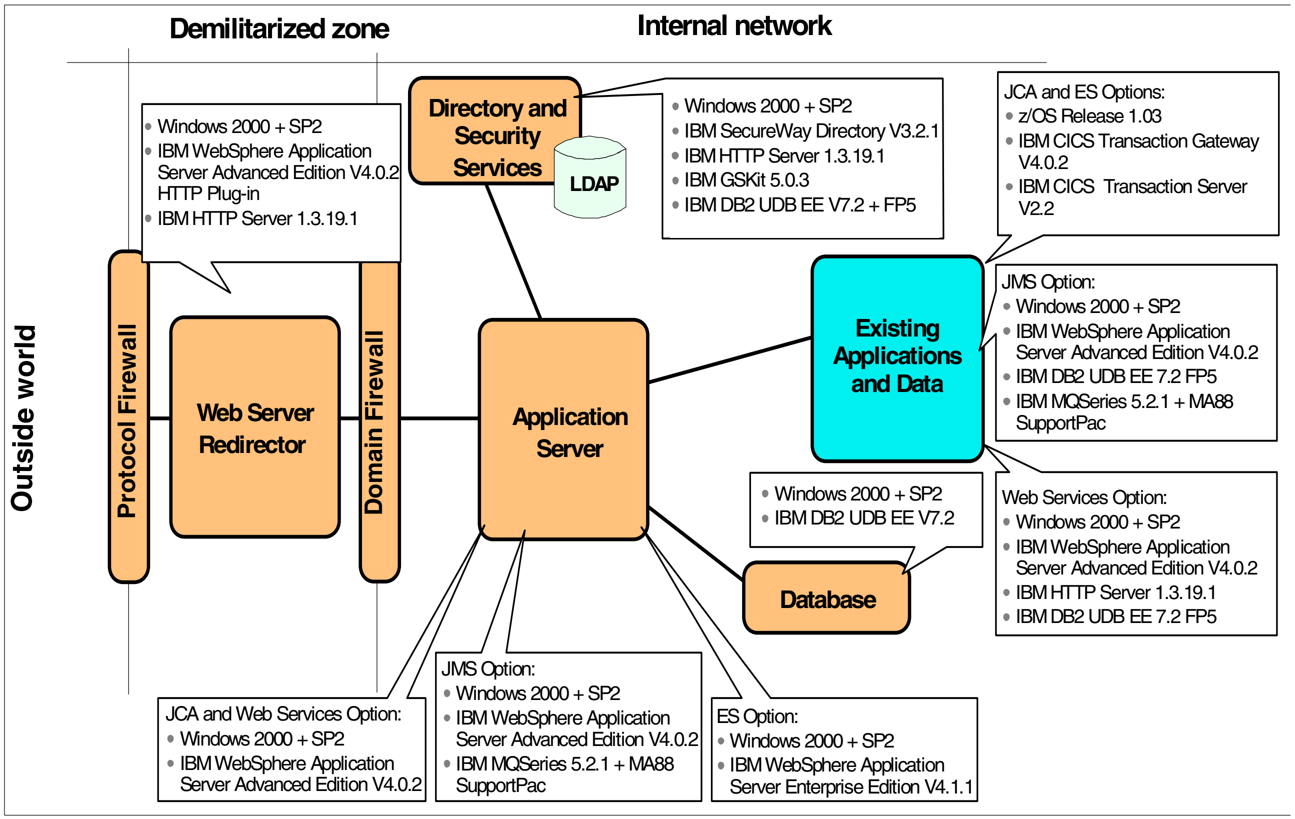

Demilitarized zone
Internal network
Outside world
Protocol Firewall
Domain Firewall
Web Server Redirector
Application Server
Directory and Security Services
LDAP
Existing Applications and Data
Database
• Windows 2000 + SP2
• IBM WebSphere Application Server Advanced Edition V4.0.2 HTTP Plug-in
• IBM HTTP Server 1.3.19.1
• Windows 2000 + SP2
• IBM SecureWay Directory V3.2.1
• IBM HTTP Server 1.3.19.1
• IBM GSKit 5.0.3
• IBM DB2 UDB EE V7.2 + FP5
JCA and ES Options:
• z/OS Release 1.03
• IBM CICS Transaction Gateway V4.0.2
• IBM CICS Transaction Server V2.2
JMS Option:
• Windows 2000 + SP2
• IBM WebSphere Application Server Advanced Edition V4.0.2
• IBM DB2 UDB EE 7.2 FP5
• IBM MQSeries 5.2.1 + MA88 SupportPac
• Windows 2000 + SP2
• IBM DB2 UDB EE V7.2
Web Services Option:
• Windows 2000 + SP2
• IBM WebSphere Application Server Advanced Edition V4.0.2
• IBM HTTP Server 1.3.19.1
• IBM DB2 UDB EE 7.2 FP5
JCA and Web Services Option:
• Windows 2000 + SP2
• IBM WebSphere Application Server Advanced Edition V4.0.2
JMS Option:
• Windows 2000 + SP2
• IBM WebSphere Application Server Advanced Edition V4.0.2
• IBM MQSeries 5.2.1 + MA88 SupportPac
ES Option:
• Windows 2000 + SP2
• IBM WebSphere Application Server Enterprise Edition V4.1.1

Figure 1-8 Directly Integrated Single Channel application pattern::Product mapping

1.2.5 Guidelines and related links

The Application patterns, Runtime patterns, and product mappings are intended to guide you in defining the application requirements and the network layout. The actual application development has not been addressed yet. The Patterns Web site provides guidelines for each Application pattern, including techniques for developing, implementing, and managing the application based on the following:

- *Design* guidelines instruct you on tips and techniques for designing the applications.
- *Development* guidelines take you through the process of building the application, from the requirements phase all the way through the testing and rollout phases.
- *System management* guidelines address day-to-day operational concerns, including security, backup and recovery, application management, and so forth.
- *Performance* guidelines give information on how to improve the application and system performance.

1.3 Summary

The IBM Patterns for e-business are a collective set of proven architectures. This repository of assets can be used by companies to facilitate the development of Web-based applications. They help an organization understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented.

2

Introduction to products

This chapter describes the various products that are discussed in this redbook. Clearly the emphasis is on Lotus Domino and WebSphere Application Server, but the use of other products can greatly enhance functionality or provide new ways of integrating these products with an existing infrastructure.

The first part of this chapter presents a high-level overview of the IBM software portfolio; the second part provides details about the specific products used in this redbook.

2.1 The IBM software portfolio

The extensive IBM software portfolio is organized into four distinct brands covering different sets of functionality. Solutions built using IBM software will typically require products from more than one brand. For example, a Web application may need a Web application server (WebSphere), a relational data store (DB2®), a workflow engine or collaboration features (Lotus) and a tool to enable single sign-on (Tivoli®). Clearly this is just one of many examples, which demonstrates that while this book focuses on how to integrate just two products, other products must also be considered for a holistic view.

The portfolio is designed to be open, and therefore interchangeable with other vendors' products. This "plug and play" architecture relies on the strict adherence to open standards to allow, for example, an IBM directory solution to be interchanged with another vendor's solution that is standards-compliant; in this case a standard called LDAP or lightweight directory access protocol.

Another important factor in the IBM software strategy is that its software runs on many different platforms. This allows the customer to choose software on a project requirements basis instead of being limited by hardware platform support. An important benefit is that development and testing work can be done on smaller and cheaper platforms before the software is moved onto more scalable/reliable platforms that require significant investment. Consult the following Web sites for a comprehensive view of the IBM software portfolio:

```
http://www.ibm.com/software
http://www-3.ibm.com/software/info/referenceguide/G3252132-00_lowres.pdf
```

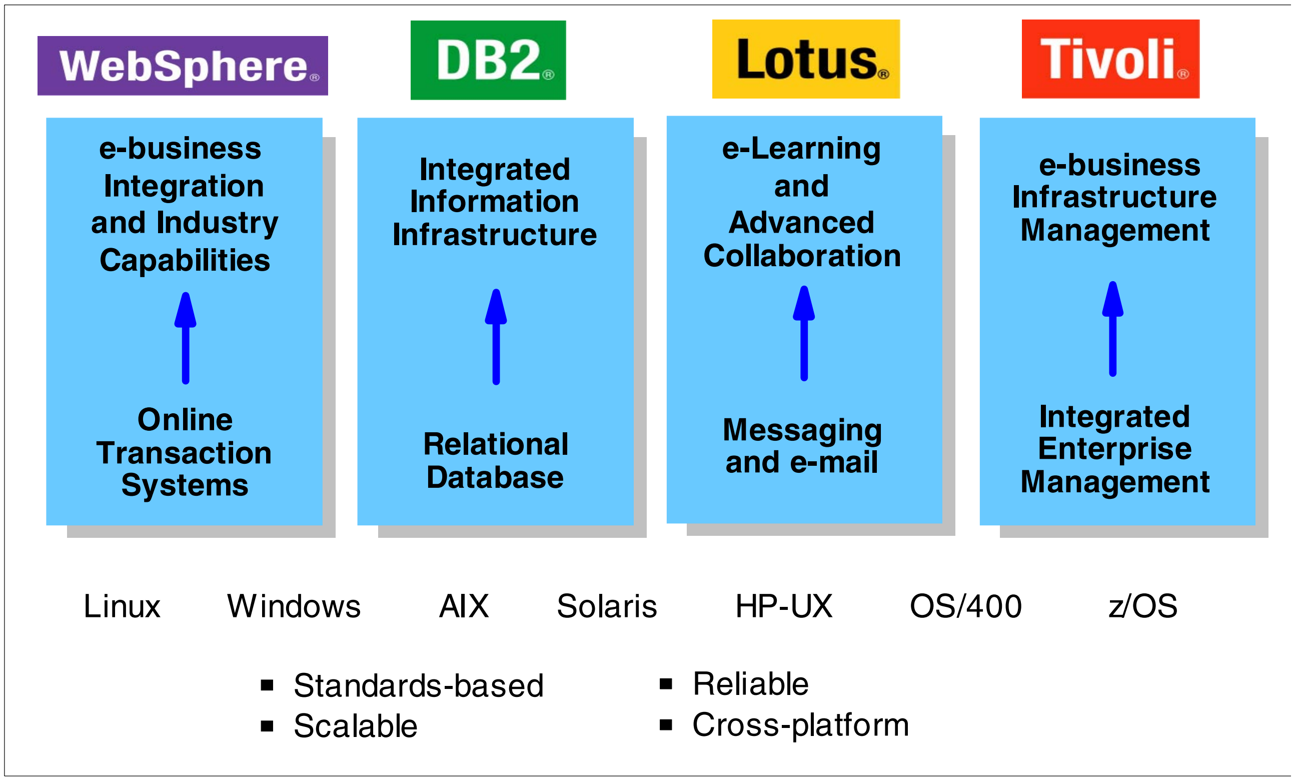

WebSphere®
DB2®
Lotus®
Tivoli®
e-business Integration and Industry Capabilities
Online Transaction Systems
Integrated Information Infrastructure
Relational Database
e-Learning and Advanced Collaboration
Messaging and e-mail
e-business Infrastructure Management
Integrated Enterprise Management
Linux
Windows
AIX
Solaris
HP-UX
OS/400
z/OS
Standards-based
Scalable
Reliable
Cross-platform

Figure 2-1 The IBM software brands and their strengths

2.1.1 WebSphere

The WebSphere brand includes software that falls into three main focus areas.

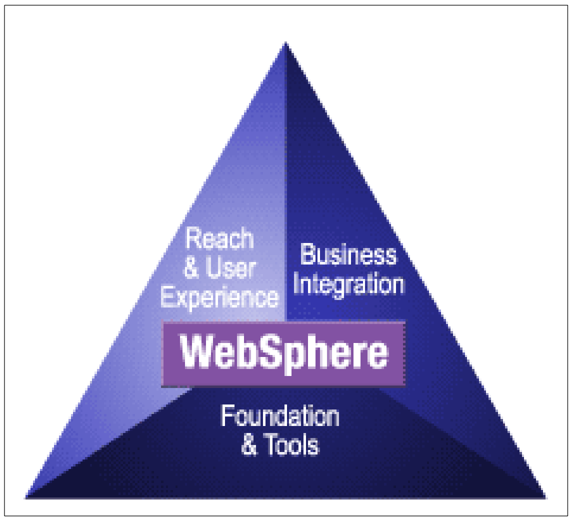


Figure 2-2 The WebSphere pyramid

Foundation and tools

The foundation products include the WebSphere Application Server and WebSphere Studio families. The application server is a Web application server based on the Java 2 Enterprise Edition (J2EE) standards. WebSphere Studio is a set of tools to rapidly build J2EE applications to run on WebSphere Application Server or any J2EE-compliant application server. These products are discussed in much more detail later in this chapter. Additional information is on the Web at:

`http://www-3.ibm.com/software/webservers/appserv/`

Business integration

The WebSphere Business Integration strategy is aimed at helping organizations integrate applications and processes quickly, without prohibitive cost and complexity. There are two types of integration:

- **Application connectivity**

  Software in this category links the various internally developed, legacy, and packaged applications that are common in a business environment. These isolated islands of applications typically have incompatible data formats and communication protocols. Application connectivity makes the output of these diverse applications available whenever and wherever needed. The suite of products includes key connection technologies such as message-based middleware, pre-built adapters for most key application packages, and Web services for advanced, universal connectivity. This is based on the WebSphere MQ family of products.

- **Business process integration**

  This middleware helps companies improve business performance by evolving from an environment of manual business processes to automated ones. Products in this section include WebSphere MQ Workflow and Crossworlds®.

Reach and end-user experience

This group of products focuses on the interface with users, no matter what device they connect with.

- **WebSphere Portal**

  Software in the WebSphere Portal family provides a single point of interaction with information applications, processes, and people to help build business-to-employee (B2E), business-to-business (B2B), and business-to-consumer (B2C) portals. WebSphere Portal also supports a wide variety of pervasive devices, enabling users to interact with their portal anytime, anywhere, and using any device—wired or wireless.

- **WebSphere Commerce**

  WebSphere Commerce provides solutions for sell-side e-commerce, customer relationship management, and partner relationship management. The product also includes some buy-side functionality, but does not focus on this area. The comprehensive set of integrated software components help sellers build, maintain, and manage sites or stores to sell goods and services on the Web in both business-to-business and business-to-consumer markets.

- **Pervasive**

  The WebSphere Everyplace™ family of products provides a comprehensive product line for voice portal, wireless access, contact center, and service provider markets.

More information on WebSphere software can be found at:

`http://www.ibm.com/websphere`

2.1.2 DB2

The DB2 brand is mainly known for its DB2 relational database, but the brand is also focused on managing unstructured data from various sources, including video, images, and sound, as well as desktop productivity documents. Reporting on trends in data is covered by the business intelligence area.

Business intelligence

Business intelligence delivers data warehousing, data mining, and OLAP solutions to spot customer trends, create customer loyalty, enhance supplier relationships, reduce financial risk, and uncover new sales opportunities.

Content manager portfolio

This provides the enterprise content management infrastructure to store, access, and manage the full spectrum of digital information generated in e-business. Large collections of scanned images, facsimiles, electronic office documents, XML and HTML files, computer output, audio, and video can be managed.

DB2 product family

These include database engines and data integration tools delivering multi-media and Web-ready data management for all types of business applications.

IMS™

This is the IBM transactional and hierarchical database management system for on-line operational and e-business applications and data.

Data management tools

Data management tools are specifically designed to enhance the performance and operations of IMS and DB2 for OS/390®. They include a comprehensive portfolio of affordable, high quality tools that add up to increased ease of use, less training time, and higher productivity for DBAs.

Information integration
Information integration is about integrating diverse forms of data across and beyond the enterprise. Business drivers such as increasing customer loyalty, improving operational efficiency, and quickly capitalizing on emerging opportunities all require rapid and coherent information integration.

Informix® product family
From data warehousing, analysis, and decision support, to Web content delivery, Informix products are engineered to enable today's businesses to efficiently manage any kind of information, anywhere, and at any time.

U2 product family
Extended RDBMS offering high performance, scalable data management environments for embedding in vertical applications.

More information on IBM database and data management products is on the Web at:

`http://www-3.ibm.com/software/data/database/`

2.1.3 Lotus

Messaging and Wireless
The core product in this category is Lotus Domino, which is covered in much more detail in 2.2.1, "Lotus Domino 6.0" on page 17. Domino is a multi-platform foundation for collaboration and e-business. Domino-based solutions range from Enterprise Messaging and Calendar and Scheduling, to a variety of collaborative applications such as CRM, HR, document libraries, team discussion databases, and much more. The application server platform has its own development and administration tools and also includes other pieces of infrastructure, such as directory and security services. It is this all-in-one approach that has allowed Notes and Domino to be extremely successful in both the small to medium marketplace and in the enterprise customer segment.

Domino applications can be accessed via a variety of clients, ranging from the dedicated Lotus Notes client for Windows or Mac to Web browsers, PDA devices, mobile phones, and even Microsoft Outlook for mail. The various client options have their pros and cons, the discussion of which is beyond the scope of this high level description.

Support for mobile devices comes with the Domino Everyplace suite of products, which allows for simple access to applications through mobile phones, and offline synchronization on the major PDA platforms using the Mobile Notes™ Client.

Domino supports many of today's industry standard protocols, and has often helped drive these standards from their initial development.

Advanced collaboration
This section covers a wide variety of products which enable users to work together more effectively and share unstructured content. Highlights of the products in this area include:

Sametime® focuses on the real-time aspect of collaboration by providing instant messaging, application sharing tools, and an interface to identify when other users are online. This presence information has been fully incorporated into the other applications in the Lotus portfolio. This product is covered in more detail in the next section.

QuickPlace™ allows users to collaborate asynchronously in a team/project space that gives them access to tools such as project calendars, project to-do lists, document repositories, and discussion areas. This is all set up and maintained with little or no input from IT, allowing the users to manage their own environment.

Domino.Doc® is an enterprise document management system built on top of the Domino platform. This allows users to manage document versions and automate document-driven processes like review and approval, assembly and publishing, archiving, and records management.

Lotus Workflow™ extends the workflow capabilities of Domino to simplify the development of mission-critical workflow applications that can be supported and maintained, rolled out across the enterprise, and quickly modified as processes evolve. By providing point-and-click tools and reusable object libraries for routing rules, role assignment, deadline handling, and task automation, even complex processes can be automated with little or no programming.

Discovery Server identifies relationships among documents, people, and topics, allowing users to quickly find the information and expert help needed to take fast, effective actions. It builds on top of Sametime to instantly link discovered people through online awareness and instant messaging. The Lotus Discovery Server™:

- Searches for information quickly, across an entire enterprise, to help manage information overload
- Identifies the right expertise, allowing skills across departmental and geographical boundaries to be mapped
- Captures and catalogues information and intellectual assets for reuse to avoid recreating work

IBM Lotus Extended Search is a scalable, server-based technology that searches across many parallel content and data sources and returns query results into either a Notes or a Web application.

With Lotus Extended Search, you can:

- Find all the information needed from one easy-to-use interface
- Easily integrate the search into other Domino or Web applications
- Search in parallel across Notes domains, legacy databases, and popular Web search sites
- Get aggregated results presented as a single, ranked hit list
- Save, re-use, and share searches
- View shared searches to identify people with similar interests
- Start quickly with ready-to-use Notes and Web client applications

e-Learning

The e-Learning products allow organizations to build and deliver a variety of e-learning programs that blend different types of learning. These include live classroom environments using Virtual Classroom, and native support for self-paced, collaborative learning using LearningSpace® 5. The e-learning products let you incorporate content from almost any source, and provide comprehensive tracking and reporting capabilities.

More information on the Lotus suite of products can be found at:

http://www.lotus.com

2.1.4 Tivoli

The Tivoli brand focuses on the management of the infrastructure required for e-business. The products can be grouped in five functional areas.

Performance and availability management

IBM Tivoli performance and availability management solutions constantly gather information on hardware, software, and network devices, and, in many cases, cure problems before they actually occur. This technology identifies critical problems, as well as misleading symptoms and effects, and then either notifies support staff with the appropriate response, or automatically cures the problem.

Configurations and operations management

This suite of applications distributes software reliably and securely, manages the change and control of IT assets, automates workflow through the enterprise, and remotely controls systems and applications

Security management

IBM Tivoli security management solutions address two e-business challenges: automated identity management and security event management. The IBM Tivoli identity management solution helps by bringing users, systems, and applications online fast, while effectively managing users, access rights, and privacy preferences throughout the identity lifecycle. The IBM Tivoli security event management solution helps actively monitor, correlate, and respond to IT security threats across your e-business.

Tivoli storage management

Storage management protects an organization's data from hardware failures and other errors by storing backup and archive copies of data on offline storage. It scales to protect thousands of computers running a dozen OS platforms. Tivoli storage products provide a combination of scalability, intelligent data technology, disaster preparation, and broad platform and application support, all through one centralized, automated solution.

zOS and OS/390 management

The Tivoli suite of systems management products for z/OS™ and OS/390 can manage large heterogeneous environments as a single entity to ensure an entire infrastructure is reliable, available, and efficient. At the same time, intelligent Tivoli management products help you effectively manage the business processes that your technology has been designed to support.

More information on Tivoli products can be found at:

```
http://www.tivoli.com
```

2.2 Products used in this book

In this section we provide details about the products we used to prepare this redbook.

2.2.1 Lotus Domino 6.0

Lotus Domino is a comprehensive application platform for collaboration that handles both connected and disconnected requirements for data and applications. Some customers initially purchase Lotus Domino for its built-in enterprise e-mail and calendar & scheduling, making those types of applications the most widely deployed collaborative applications. However, many customers also exploit the “more than mail” capabilities that support core business processes, enabling employees to work together efficiently and securely. Lotus Domino is comprehensive because it provides the complete infrastructure needed to create, test, deploy, and manage distributed, multi-lingual applications—including directory, database, application server, administration, security, connectivity, Web server, e-mail server, calendaring engine, and so on—all in one application. This makes it ideal for the small-to-medium business since it allows for powerful applications to be built with minimal infrastructure.

Domino developers can design applications for the Lotus Notes client, Web browsers, mobile phone and handheld devices, or most commonly, for a hybrid environment accessed by multiple types of clients. Hybrid client Domino applications can use replication and off-line services to produce secure synchronized applications that work as well in a disconnected mode as when accessed on a server over a network. Replication enables users to save a local copy of a Domino application and its data on a file system and to periodically synchronize the data, so users can be productive and efficient even when disconnected from the network. If Domino Off-Line Services (DOLS) is added to a browser client, then users can work with Domino Web applications in a disconnected state with similar productivity results.

Examples of Domino solutions include document-centric and workflow process routing, such as project teamrooms, document repositories, discussion forums, sales force enablement, and employee self-service applications. Businesses of all sizes have benefited from Domino applications.

For descriptions of some real-world implementations using Domino 6, see the Lotus Web site at:

`http://www.lotus.com/success`

The value of Domino application development

As a comprehensive application platform, Lotus Domino includes a tool for rapid application development (RAD), a document-based object model, and broad programming language support for building custom collaborative applications. With these choices, your organization can leverage many developer skills to develop a Domino application. For example, if you need a collaborative application such as a discussion forum or document library, and you have only a minimal programming background, you can easily create a Domino application from a ready-made template without writing any code.

If you want to add workflow to move documents around in a role-based workflow, you can add macros and formulas to the original template. When conditional logic needs to be added to send off notifications based on the contents of a document or view of data, a developer with basic programming skills can enhance the application further. In a typical custom Domino application, very basic programming skills are sufficient for the majority of tasks.

For more advanced solutions, developers can use Java, COM, C/C++, or CORBA. The multiple interfaces of Lotus Domino to a single object model enable a developer to pick the best language for the task, reusing his skills in new applications and solutions.

Some solutions require non-Notes data, multiple languages, or globalization support. The rapid application development capabilities of Lotus Domino facilitate such solutions. Using visual data mapping techniques, a developer can easily, quickly integrate relational data with Notes data. No programming is required with technologies known as Lotus Connectors. Domino applications that require global deployment can have all elements translated to a variety of languages with some straightforward forms. All of these RAD activities take place in the integrated development environment called Domino Designer®.

The features of Domino application development in Domino 6 can benefit your organization by increasing code and design element reuse, simplifying team development, and providing general enhancements and improvements over Release 5—all of which will help improve developer productivity. Major new features include:

- XML management (DOM parser, SAX parser, XSLT transformer, XML important and export utilities)
- Programmatic rich text handling
- JSP tag interface
- Team development features, like design element locking
- Tighter integration with external systems, such as relational databases
- Dynamic Web page creation tools

As you can see, Lotus will continue to enhance the Lotus Notes and Domino application model with new features as required by customers. In coming releases of Lotus Notes and Domino, you can expect Lotus to stay consistent with its open standards platform.

The Domino roadmap for application development builds on the fundamental premise that Lotus Domino is a flexible and open platform, as demonstrated by XML and broad programming language support. Flexibility and openness are key to a Domino application's ability to leverage J2EE. You can extend your Domino application investment in data and application logic, exposing relevant applications and data using Web services (through LotusScript or Java) or JSP tags to integrate with next generation or WebSphere applications.

Along with the progressive innovation that Lotus will add to Domino Designer and to the Domino programming model, there will be continued integration with WebSphere Studio. This integration will not replace Domino Designer, but will help facilitate teams of developers using both Domino and WebSphere Application Server in their environment. Such teams will benefit by utilizing the strengths of each system when building applications.

Domino applications and Web services

Web services is a collection of emerging standards that simplifies application integration by providing a standardized access protocol known as Simple Object Access Protocol or SOAP. Developers can express the application interface using XML. Today, Domino developers can add Web services interfaces to existing Domino Release 5 applications on any platform, using about 35 lines of LotusScript or Java code. The only tools required to use Web services in your Domino applications are a Lotus Domino server and Domino Designer. Samples are available at:

`http://advisor.com/Articles.nsf/aid/DEVEGO2`

You can download a toolkit from:

`http://www.alphaworks.ibm.com`

Lotus Domino can host Java-based Web services that expose Domino data and functionality. Using a combination of a J2EE server like WebSphere, appropriate SOAP classes, and the Domino Java Objects, developers can expose desired portions of their current Domino applications as Web services. To do so requires some Java development skills and knowledge of the Domino object model. The IBM WebSphere Application Server ships with the needed SOAP classes, and the WebSphere Studio development environment has wizards for creating, consuming, managing, and deploying Web services. This helps improve developer productivity and lets developers focus on exposing the essential parts of a Domino application as the Web services. This is covered in more detail in Appendix A, “Web services” on page 171.

Domino and J2EE

One of the key differences between the J2EE application server and the Domino application server is that Lotus Domino provides a fully integrated environment. It handles nearly all aspects of the application environment from application execution to user authentication, directory services, data hosting, and presentation display all in one system.

In contrast, the J2EE model has elements for providing the application with all the same information, but the J2EE server is not responsible for fulfilling all the aspects that the Domino server fulfills. The J2EE server calls out to different parts of the customer environment to fulfill the requests of data, directory information, and so on. For example, while the J2EE specification outlines how the server can get data from a data store into the application using JDBC, J2EE does not require that the server contain the database manager itself.

More information on Lotus Domino can be found at:

`http://www.lotus.com/domino`

2.2.2 WebSphere Application Server 5.0

J2EE applications and application development model

Traditionally, J2EE applications use a tiered application model. The key elements of an application are divided into distinct components that perform their specific functions in an environment separate from the other elements. The simplest tiered model is broken up into presentation logic, application logic, and data housing. Each of these tiers performs a specific function of the application and can reside on one or more systems. An enterprise can deploy application elements to any type of hardware and location for the maximum benefit of the business.

This tiered model provides valuable flexibility when building, deploying, and reusing applications. Based on this type of architecture, an application’s presentation logic tier can be located geographically near the users on inexpensive hardware; with the application logic tier located at a corporate headquarters on a more powerful, expensive system; and the data housing tier on a mainframe system that, perhaps, has been running the business for decades. This example is quite extreme, but possible using the J2EE tiered application model. Another benefit of this model is that it allows an application’s elements to be easily reused to benefit some other part of the business.

As an example, consider an inventory application built around this model that lets manufacturing know how many widgets are in the warehouse and how many should be made based on sales forecasts. The sales force also needs an application to know how many widgets are in the warehouse and what the production schedule is for manufacturing, so it can set customer expectations for filling orders. Instead of the new sales application duplicating the functionality of the manufacturing application, the manufacturing applications logic can be reused and tied into the new sales application.

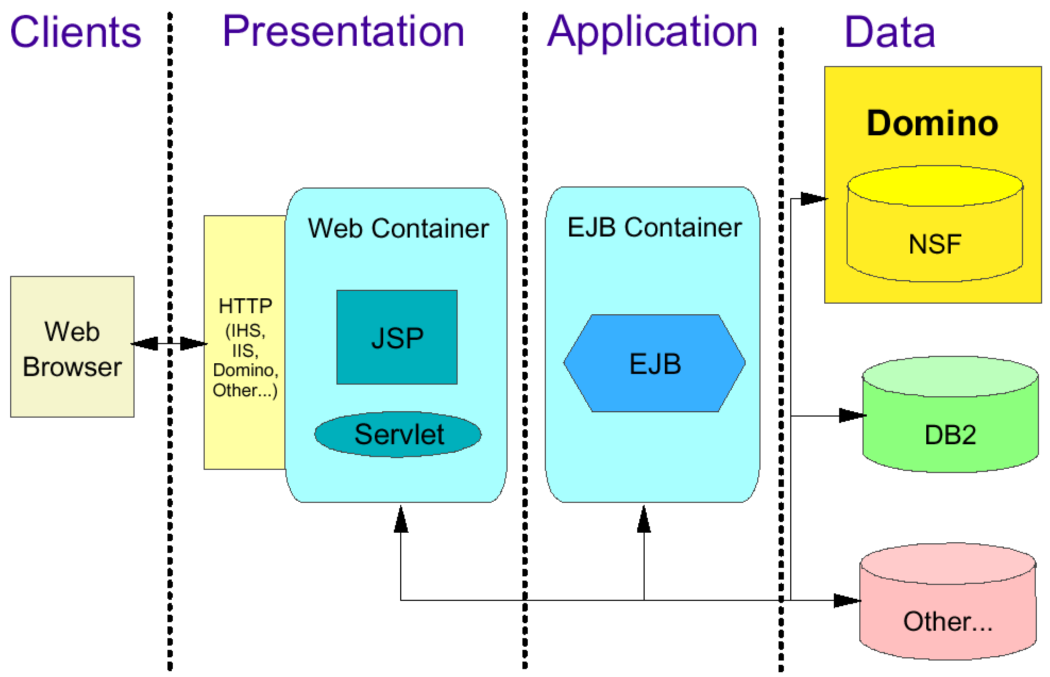


Figure 2-3 Tiered application model

About WebSphere Application Server

IBM WebSphere Application Server, Version 5.0 is a comprehensive Java 2 platform, Enterprise Edition (J2EE)1.3 and Web services technology-based application server that integrates enterprise data and transactions with the rest of an organization's infrastructure. In this rich application deployment environment, it is possible to build, manage, and deploy dynamic e-business applications, handle high-transaction volumes, and extend back-end business data and applications to the Web.

WebSphere increases developer productivity through its close interlock with IBM WebSphere Studio, a tightly integrated Java development environment based on open Eclipse workbench technology. This allows employees to develop, test, and deploy Java and Web services applications with easy access and minimal errors. The integrated development and deployment platform of WebSphere Application Server optimizes development resources through its ability to reuse CORBA, C++, Java, and legacy assets.

Application adapters that quickly and easily extend enterprise applications to e-business also help you make use of current resources. WebSphere Application Server helps reduce the risk, complexity, and cost of using and deploying application adapters through its support for J2EE Connector Architecture (JCA). JCA gives a consistent way of connecting to and communicating with a wide range of enterprise systems and applications without the need for advanced programming skills or extensive coding. This allows for reuse, and integration with disparate systems and applications.

Increase productivity with messaging integration

WebSphere Application Server enables dynamic application interaction through a native, high-performance Java Message Service (JMS) provider based on IBM WebSphere MQ technology and support for J2EE 1.3, Enterprise JavaBeans (EJB)2.0. The JMS application programming interface (API) increases productivity by defining a common set of messaging concepts and programming strategies. JMS further simplifies development by allowing loosely coupled, reliable asynchronous interactions among J2EE components and legacy systems capable of messaging. EJB 2.0 message beans and container-managed messaging save valuable programming time and skill by allowing requests to be processed without requiring code to check for messages when they arrive. Developers can easily incorporate new behavior in a J2EE application with existing business events by adding a new message-driven bean to operate on specific business events.

WebSphere Application Server and Web services

WebSphere Application Server extends the J2EE 1.3 programming model by providing support for the production-ready deployment of Web services-based applications. It builds, publishes, and manages integration-ready application services that can be used by other internal or external organizations or platforms. WebSphere Application Server supports key Web services open standards, including Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI) and Web Services Description Language (WSDL). It allows for the deployment and consumption of Web services with a variety of communication protocols, including SOAP and HTTP, JMS, or Remote Method Invocation Internet Inter-ORB Protocol (RMI /IIOP). It also allows for the administration of virtually any Web service, whether developed with Java technology or Microsoft.NET.

Web services are covered in more detail in Appendix A, “Web services” on page 171.

WebSphere Application Server Network Deployment provides extended Web services support with a private IBM UDDI Registry and IBM Web Services Gateway. The UDDI Registry, which acts as a directory of services to help users find information about Web services, enables developers to publish and test their internal e-business applications in a security-rich, private environment. Web Services Gateway helps reduce development costs by making selected services available to different divisions within an enterprise, or to customers and partners who use different protocols or are outside the firewall. Using Web Services Gateway, developers and IT managers can safely externalize a Web service so users can invoke it from outside the firewall.

WebSphere Application Server administration

WebSphere Application Server provides a browser-based interface to administer, deploy, and manage applications with ease. With WebSphere Application Server Network Deployment, these capabilities are extended to help manage configurations that include large numbers of servers. Automated application server management functions help enhance productivity and reduce administrative costs. As your business needs change, WebSphere Application Server can help you quickly and easily move from one configuration to another by providing a single browser-based administration interface across all deployment options. To let you effectively manage your operations and applications, WebSphere Application Server provides installation and administration capabilities through exposed Java Management eXtension (JMX) interfaces and an extended command-line interface. Support for JMX allows third-party products (like Tivoli software) to read and manage WebSphere software in a standard way.

WebSphere Application Server Network Deployment distributes workload across multiple servers through sophisticated load-balancing and clustering capabilities, including automatic failover capability, content-based routing to deliver more effective session management, and enhanced edge-based caching capabilities. The sophisticated load balancing and edge-of-network components provide a scalable solution for load-balancing requests between HTTP, FTP, or other TCP-based servers. Customer-defined rules and requirements can be incorporated to help the load balancer re-route requests intelligently.

WebSphere Application Server and security

WebSphere Application Server offers a sophisticated security infrastructure and extensive support of open standards-based Java specifications, including:

- Java authentication authorization services (JAAS) authenticates new principals and manages privileged information for a principal.
- Java 2 Security Model helps secure system resources.
- Java Secure Socket Extension (JSSE) helps secure communication channels based on transport level security (TLS/SSL).
- Java Cryptographic Extension (JCE) provides a framework for security encryption and message authentication.
- Java Cryptographic Architecture (JCA) provides Java cryptographic extensibility, as with PKI integration.
- Common Secure Interoperability (CSIv2) supports secure interoperability between application services Sophisticated enterprise topologies and infrastructure can be implemented through WebSphere pluggable security architecture. These include:
  - Pluggable user registries to enable you to exploit LDAP or custom registries
  - Web single sign-on exclusively provided with WebSphere software or through integration with front-end authentication end-points with Trust Association Interceptor (TAI) technology
  - Highly secure access to enterprise information systems through a pluggable principal and credential mapping facility

WebSphere Application Server Enterprise

An extended J2EE and Web services, build-to-integrate platform to create, compose, and choreograph dynamically adaptable networked application flows and behaviors.

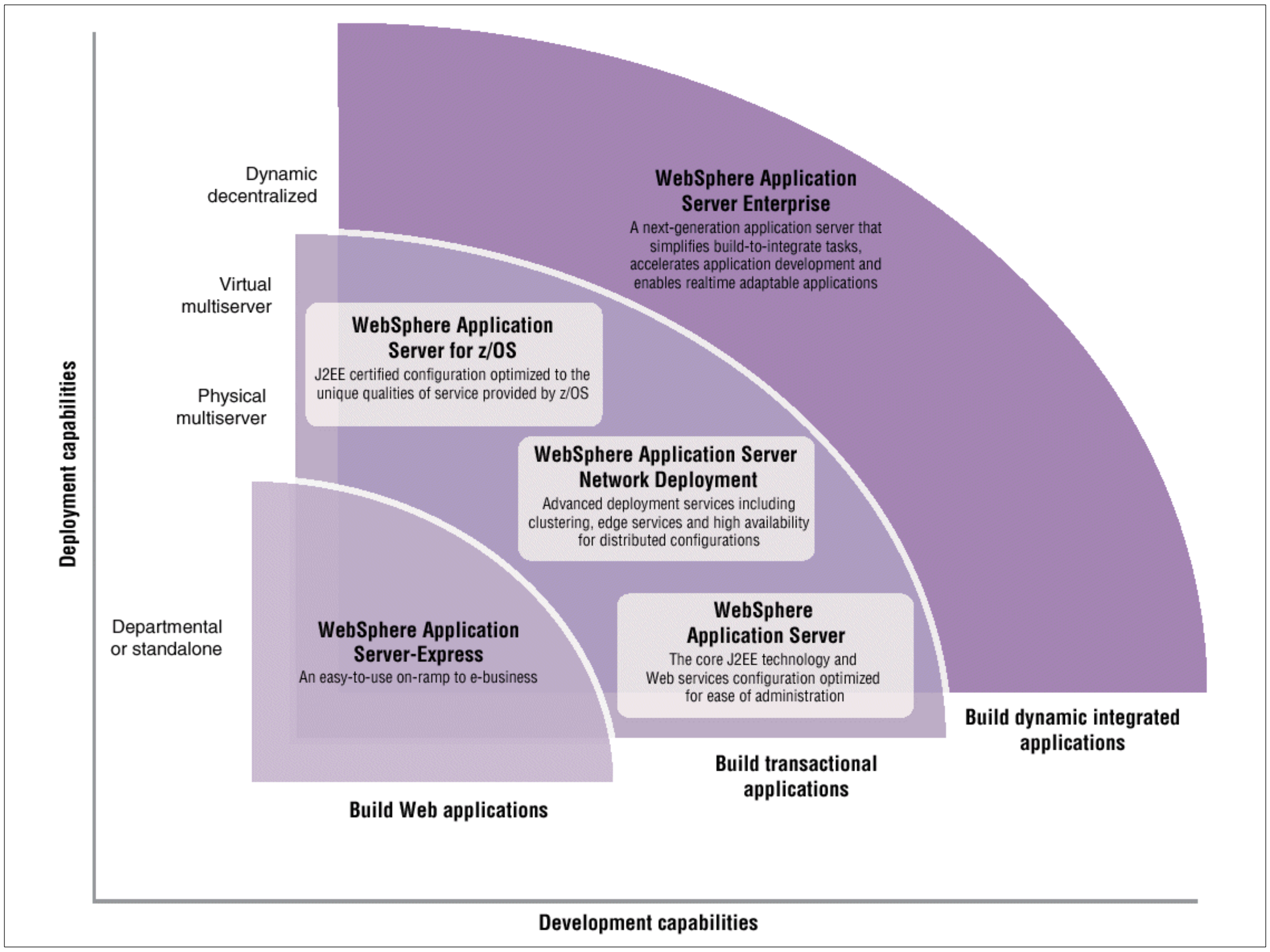

Deployment capabilities
Dynamic decentralized
Virtual multiserver
Physical multiserver
Departmental or standalone
WebSphere Application Server Enterprise
A next-generation application server that simplifies build-to-integrate tasks, accelerates application development and enables realtime adaptable applications
WebSphere Application Server for z/OS
J2EE certified configuration optimized to the unique qualities of service provided by z/OS
WebSphere Application Server Network Deployment
Advanced deployment services including clustering, edge services and high availability for distributed configurations
WebSphere Application Server
The core J2EE technology and Web services configuration optimized for ease of administration
WebSphere Application Server-Express
An easy-to-use on-ramp to e-business
Build Web applications
Build transactional applications
Build dynamic integrated applications
Development capabilities

Figure 2-4 The WebSphere family of application server products and their capabilities

More information on WebSphere Application Server can be found at:

http://www-3.ibm.com/software/webservers/appserv/

2.2.3 Lotus Sametime 3.0

The IBM Lotus Sametime family includes three components: the Sametime server, the Sametime Connect client, and the developer toolkits.

- The server provides the platform that manages the flow of information between the Sametime Connect clients, including text messaging, streaming audio and video, a shared whiteboard, and shared applications.
- Users use either the Sametime Connect client or a browser-based equivalent to exchange ideas and present information through instant messaging via the Sametime Server.
- While Sametime is an application in its own right, the developer toolkits give organizations a way to embed real-time collaboration into a wide range of Web- and Windows-based applications, thus encouraging collaboration related to the application, or contextual collaboration.

These three components help people collaborate and communicate effectively with others in real time, either across the hall or around the world.

Examples of the value of the application include:

- Sales reps with instant, immediate access to information and people back at the office close more sales faster.
- A place to share documents with colleagues around the world in real time reduces travel expenses and makes decisions happen faster.
- Communicating directly and securely with customers improves customer satisfaction and gives an enterprise a competitive advantage.

Three additional components help companies expand the reach of Sametime by improving scalability and connecting with Sametime users outside a company's local area network:

- **Sametime IM Gateway** gives companies a way to securely connect to other instant messaging communities.
- **Enterprise Meeting Server** allows companies to host and administer enterprise-wide Web conference environments.
- **Sametime Everyplace** gives companies a way to bring presence awareness and the instant messaging capabilities of Sametime to users with mobile phones and wireless PDAs.

Presence awareness, instant messaging and Web conferencing are the keys to improving enterprise-wide collaboration. These three components combine with a robust, secure and scalable platform to help companies be responsive and adaptable.

Presence awareness gives a team member the ability to see if other members are online. By adding support for the SIP (Session Initiation Protocol) for Instant Messaging and Presence Leveraging Extentions, or SIMPLE, presence awareness extends securely beyond the firewall to a supplier's or partner's instant messaging community. With Sametime, presence awareness isn't just about who is connected, it also indicates if a person is connected via a mobile phone or wireless PDA through Sametime Everyplace.

Instant messaging makes sending and receiving text messages in real time a fast, efficient way to communicate. The Sametime Connect instant messaging client is about much more that exchanging text. From the Connect client, people can start an ad hoc Web conference to exchange richer information or share a document or application. Because Sametime can be customized and integrated with other enterprise applications, instant messaging can also be a way to get information, such as directory information, from enterprise applications.

Web conferencing broadens the kind of information team members exchange through Sametime. Streaming audio and video, shared documents, presentations and application sharing make Sametime a robust platform for presenting information. Application sharing has many uses—from presenting information to providing technical support. A person can even hand off control of an application to another and reclaim control as needed. Sametime automatically converts popular file types into whiteboard pages.

More information on Sametime can be found at:

`http://www.lotus.com/sametime`

2.2.4 Tivoli Access Manager for e-business

IBM Tivoli Access Manager for e-business provides standards-based authentication, single sign-on, and authorization services for Web pages and Web applications. Access Manager's security services ensure that only authorized users are granted access to data, services and transactions. It provides integrated security for J2EE security implementations, as well as out-of-the-box support for many key Web vendors' portal, ERP, CRM, and other applications.

Access Manager for e-business allows a combination of architectures to be accommodated in an Access Manager-protected environment, for example:

- The highly secure, highly scalable, and highly manageable security layer approach via the WebSEAL proxy
- The WebSphere Edge Server's Caching Proxy, configured to use an Access Manager plug-in
- Web server plug-ins (for Microsoft IIS 5.0 on Windows, iPlanet 6.0 on Solaris 7, and Apache)

Access Manager allows Web application programmers to focus on business logic and to link their applications with its security infrastructure using Java standardized security calls. There is specialized support for WebSphere Application Server that enables container-based authorization rules to be managed by Access Manager, along with other, non-WebSphere objects in the protected object namespace.

Support for sending credential attributes to back-end servers makes it easier to integrate applications with Access Manager. Web Portal Manager (an administration tool offering robust, delegatable, Web-based administration) and Web e-Community Single Sign-On (offering a Master Authentication Server that provides single sign-on across domains) provide integrated policy-based security management for the extended enterprise; enabling customers, business partners, employees, suppliers, and distributors to securely access the enterprise portal resources they need to access, when they need to access them, in a trusted fashion.

There is also support for Microsoft environments, with support for Microsoft Active Directory and NTLM and Kerberos-based single sign-on; SSO to Microsoft IIS can provide access to your Tivoli Access Manager-protected applications and resources.

Tivoli Access Manager product information can be found at:

http://www.tivoli.com/products/index/access-mgr-e-bus/

2.2.5 IBM Directory Server

IBM Directory Server V5.1 provides a powerful Lightweight Directory Access Protocol (LDAP) identity infrastructure that is the foundation for deploying comprehensive identity management applications and advanced software architectures like Web services. IBM Directory Server provides the following features and benefits:

- LDAP V3 support ensures compatibility with industry standard LDAP-based applications.
- 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 including AIX®, Solaris, Microsoft Windows 2000, and HP-UX, as well as SuSE, United Linux, and Red Hat Linux distributions for Intel and IBM zSeries™ platforms.
- Compatibility with IBM environment-specific directories, Lotus Domino, OS/400®, and z/OS (OS/390).
- Robust replication capability for both master and subordinate replication, cascaded replication and peer-to-peer replication with up to dozens of master servers.
- Usability features with Web Administration GUI such as Dynamic and Nested Groups, along with Sorted and Paged Search Results.
- Password strength and ACL protection features as options for enhancing security.
- DSML V2 support extends the reach of the directory to Web services.

- Enhanced, filter-based ACL support.
- LDAP access to server configuration data.
- Tight integration with IBM operating systems, WebSphere middleware, and Tivoli identity management and security products.

More information on IBM directory server can be found at:

http://www-3.ibm.com/software/network/directory/server/

2.2.6 IBM Directory Integrator

IBM Directory Integrator is designed to tie together data residing in directories, databases, collaborative systems, applications used for human resources (HR), customer relationship management (CRM), and Enterprise Resource Planning (ERP), and other corporate applications.

It acts as a synchronization layer between a company's identity structure and the application sources of identity data, eliminating the need for a centralized datastore and therefore greatly easing the deployment of an enterprise directory solution.

It features many built-in connectors to common data sources, an open-architecture Java development environment to extend or modify these connectors, and tools to apply logic to data as data is processed.

IBM Directory Integrator can help by:

- Synchronizing and exchanging information between applications or directory sources
- Managing data across a variety of data repositories, providing a consistent directory infrastructure that can be used by a wide variety of applications ranging from security and provisioning to Web services
- Providing a consistent view of additional directory-based information, such as product and pricing data, to applications beyond traditional user identity and passwords.

More information on IBM Directory Integrator can be found at:

http://www-3.ibm.com/software/network/directory/integrator/index.html

2.2.7 WebSphere Studio

IBM WebSphere Studio helps you to optimize and simplify J2EE and Web services development by offering best practices, templates, code generation, and the most comprehensive development environment in its class. It allows developers to create J2EE and Web services applications with integrated support for Java components, EJBs, servlets, JSPs, HTML, XML, and Web services, all in one development environment.

These J2EE applications can help reduce the cost and complexity of developing multi-tier enterprise services, and can be rapidly deployed and easily enhanced as the enterprise responds to competitive pressures. For example, WebSphere Studio gives your development team access to databases and links to transaction-processing systems. Developers can use the tools they need to create, edit, and validate enterprise application archive files. The development environment in WebSphere Studio provides testing and support for IBM WebSphere Application Server, which allows you to quickly and easily create J2EE applications to fit your business needs. Your system administrators can keep track of your applications with monitoring and profiling tools that feature customizable views and comprehensive error logs, and that identify, isolate and repair glitches that can decrease performance.

The built-in unit test environment allows your developers to test their applications on the fly. Performance profiling and tracing further examine your code and application development to identify memory usage, attack memory leaks, and identify bottlenecks to help you fine-tune your application performance. WebSphere Studio lets your quality assurance team test applications early in the development cycle to save time and improve project-wide efficiency. Through the performance analyzer, you can learn which operations take the most time to execute to help you improve runtime memory usage.

The built-in Web services tools in WebSphere Studio support standards like SOAP, UDDI, Web Services Description Language (WSDL) and Web Services Inspection Language (WSIL). You can construct applications by visually composing components as services. These application services can be accessed through the Web or implemented as local Java services (JavaBeans or Enterprise JavaBeans) or legacy assets. By working with all components as services, you have the flexibility to mix and match functions from different sources, bringing innovative processes, services, and value chains to market quickly and efficiently. With IBM WebSphere Studio, your developers can use robust graphical tools to quickly and easily build custom application adapters. This allows them to reuse existing resources and save time when integrating new J2EE technology-based applications with back-end systems.

As your e-business grows, you need to be able to develop applications for mobile computing devices. WebSphere Studio lets you combine WebSphere software function and extend e-business applications with the convenience of handheld computers, personal digital assistants, and Internet-enabled mobile phones, so you can reach customers regardless of location or device.

Integration is one of the biggest challenges facing businesses today. WebSphere Studio allows you to extend your business to the Web while continuing to use your existing systems and processes. This means you can integrate your new e-business applications with your legacy or packaged applications to leverage the breadth of information and processes now available on the Web through Web services. WebSphere Studio reduces development complexity by providing application workflow to help you visually manage the flow of information between application components, Web services, and back-end systems. WebSphere Studio also simplifies integration by leveraging a services-oriented architecture that allows developers to interact with all application artifacts in the same way, regardless of their underlying implementations, providing continuity within the development team.

WebSphere Studio includes a number of products and solutions. It can be tailored to meet the specific development needs of an organization—from simple Web site creation to complex e-commerce and enterprise application development. The products that make up WebSphere Studio are identified and briefly described in Table 2-1.

Table 2-1 Benefits of WebSphere Studio products and solutions

Product/solution	**Benefits**
IBM WebSphere Studio Homepage Builder	Create and publish Web sites with ease using a user-friendly interface, easy-to-use wizards, templates, and support for popular development languages like JavaScript, Dynamic HTML, and Cascading Style Sheets (CSS).
IBM WebSphere Studio Site Developer	Build, test, and maintain dynamic Web sites, applications, and Web services using Java, JavaScript, and JavaServer Pages.

Product/solution	Benefits
IBM WebSphere Studio Application Developer	Optimize and simplify J2EE application development through best practices, templates, code generation, and a comprehensive development environment with WebSphere Studio Application Developer. WebSphere Studio Application Developer Integration Edition accelerates development by adding visual workflow to build and integrate complex applications.
IBM WebSphere Studio Enterprise Developer	Give J2EE capabilities, rapid application development, and team support to diverse enterprise application development organizations.
IBM WebSphere Studio Device Developer	Create and test applications that will be deployed on handsets and other mobile computing devices.
IBM WebSphere Studio Asset Analyzer	Maintain and extend existing enterprise assets through impact analysis and graphical application understanding.
IBM WebSphere Studio Application Monitor	Resolve performance problems with J2EE applications running on the IBM WebSphere for z/OS platform, without requiring modification to the application code.
Toolkits for IBM WebSphere Studio	Quickly and easily add new function and tools targeted at application development needs.

More information on WebSphere Studio can be found at:

http://www-3.ibm.com/software/awdtools/

2.2.8 IBM WebSphere Transcoding Publisher

Multiple formats, markup languages, device capabilities, and network constraints have, up to now, threatened to limit the promise of pervasive computing. The potential of e-business will be realized only when there is a way to bridge disparate data systems seamlessly, transcending multiple data protocols, devices, and users.

This need is being met by IBM WebSphere Transcoding Publisher, which simplifies the wireless Internet and enables universal access by dynamically adapting, reformatting, and filtering Web content and applications to make them optimally suited for mobile devices, such as phones, PDAs and pagers.

IBM WebSphere Transcoding Publisher propels your business to the wireless Internet and helps ensure that your data and applications reach customers and employees in mobile environments. Transcoding Publisher can:

- Extend existing Web content to new devices by translating it between multiple markup languages and image formats
- Streamline delivery by fragmenting data and simplifying formats so that content is provided efficiently to a variety of users despite network and device constraints
- Help customize content delivery and presentation with device, network, and end user profiles to enable more effective interaction with customers, partners, and employees

Find more information at:

http://www-3.ibm.com/software/webservers/transcoding/

3

Lotus Domino and WebSphere Application Server patterns

The focus of this chapter is on IBM Lotus Domino 6.0 and IBM WebSphere Application Server V5.0 patterns. In the overall scheme of IBM Patterns for e-business, the patterns described in this chapter are referred to as *Hybrid Runtime patterns*. Hybrid Runtime patterns are used when:

- Specific middleware components, as well as associated Business and Integration patterns, are provided for each pattern described.
- Product mappings are provided for each node in the pattern described, versus the more generic node descriptions that might be otherwise provided.

The first part of this chapter is a high level discussion of application integration options that apply specifically to Lotus Domino 6.0, WebSphere Application Server V5.0 and HTTP servers that are supported with these products. These options have changed considerably since the last major point releases of these products, so a discussion of new integration options is mandatory.

The second topic in this chapter is a discussion of typical Domino-WebSphere Hybrid Runtime patterns. We map the Hybrid Runtime patterns to applicable Business and Integration patterns. We relate all of the individual subsystems described in the scenarios to Runtime patterns in this chapter, and therefore to Business and Integration patterns. The four Business patterns and the two Integration patterns, part of the general framework for IBM Patterns for e-business, are shown in Figure 3-1.

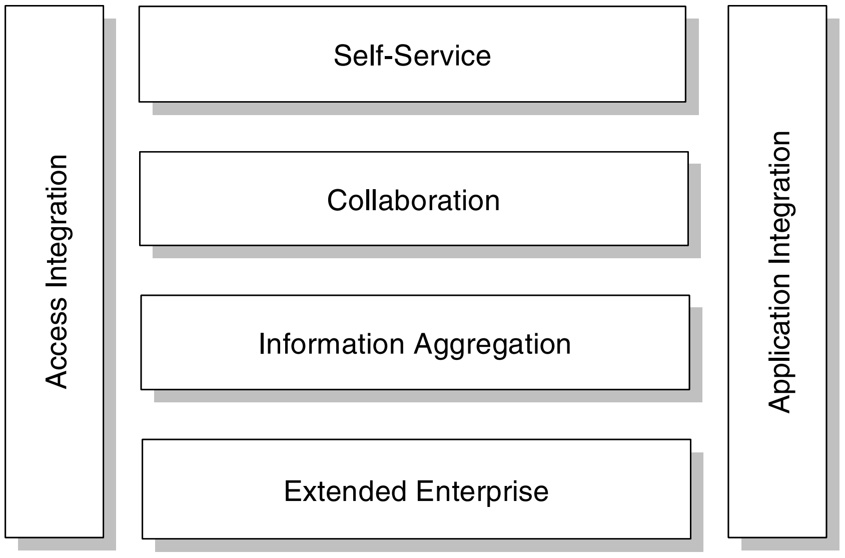


Figure 3-1 Business patterns and Integration patterns

After discussing typical Domino-WebSphere Hybrid Runtime patterns, we embark on a brief exploration of the options available from a programmatic perspective when utilizing WebSphere Application Server and Lotus Domino in a solution. These interface methods include several techniques that were previously available in Lotus Domino R5 and WebSphere Application Server 4.x, but also include new interfaces that Lotus Domino 6 and WebSphere Application Server 5.x provide.

The last topic in this chapter is the application of the IBM single sign-on (SSO) technology that can be utilized in any of the Domino-WebSphere Hybrid Runtime patterns described in this chapter. SSO is strongly recommended for any real-world applications that rely on user authentication in a multi-server environment.

3.1 WebSphere HTTP Server plug-in architecture

Before diving into specific Hybrid Runtime patterns related to Lotus Domino and WebSphere Application Server, it is appropriate to describe the connectivity options that are available when integrating these products.

The WebSphere HTTP Server plug-in architecture was introduced in WebSphere Application Server V4.0, allowing for the physical separation of the Web server from the application server. While Lotus Domino R5 included a similar capability, it was limited to the use of only Microsoft Internet Information Server (IIS) on the same physical machine as the Lotus Domino server.

The release of Lotus Domino 6 replaces the older Lotus Domino plug-in technology with the new WebSphere HTTP plug-in technology, allowing, for the first time, Domino databases to be placed behind the domain firewall, along with other sensitive data, separated from the Web server.

Note: While there is a benefit to this separation, it should be noted that the full security features of Lotus Domino 6 are still intact, and that, in fact, it is still considered safe practice to place a Lotus Domino server within the DMZ in many cases, provided normal Lotus Domino security best practices are followed.

The primary benefits of this architecture, from a Lotus Domino perspective, are:

- The elimination of denial-of-service (DOS) attacks on the Domino server itself, since it is placed behind a domain firewall.
- Better alignment with normal industry practices in n-tier application deployment, in which most persistent, sensitive data repositories are located behind a domain firewall, and not in the DMZ.

The WebSphere HTTP plug-in architecture has the following characteristics:

- The Web server plug-in is implemented as a *filter*, which examines all incoming HTTP requests and routes them to other Web servers based on the composition of the URL. Each Web server has it's own application programming interface (API) that allows filters to be implemented. For example, Microsoft IIS uses the Internet Server API Specification (ISAPI), and Lotus Domino's HTTP stack uses the Domino Server API Specification (DSAPI).
- Standards-based protocols (HTTP/S) that are supported by firewall products are used, unlike previously, when proprietary transport mechanisms (such as Remote OSE) were used.
- There are no special configuration requirements on behalf of the application servers receiving redirected requests from the plug-in; they are simply HTTP/S service providers.
- SSL can be used within the DMZ to encrypt network traffic between the Web server and the application server.
- The configuration file used by the plug-in is XML-based, and easy to administer. Multiple redirection rules may be defined to one or more application servers as dictated by the topology chosen.
- The plug-in supports load balancing and failover capabilities, which offer further scalability with very little additional administrative effort.

Some of the Hybrid Runtime patterns presented later in this chapter utilize a separate logical node with a Web server that connects to an application server using the new Web server plug-in architecture. The simple diagram in Figure 3-2 illustrates the use of the WebSphere HTTP plug-in architecture.

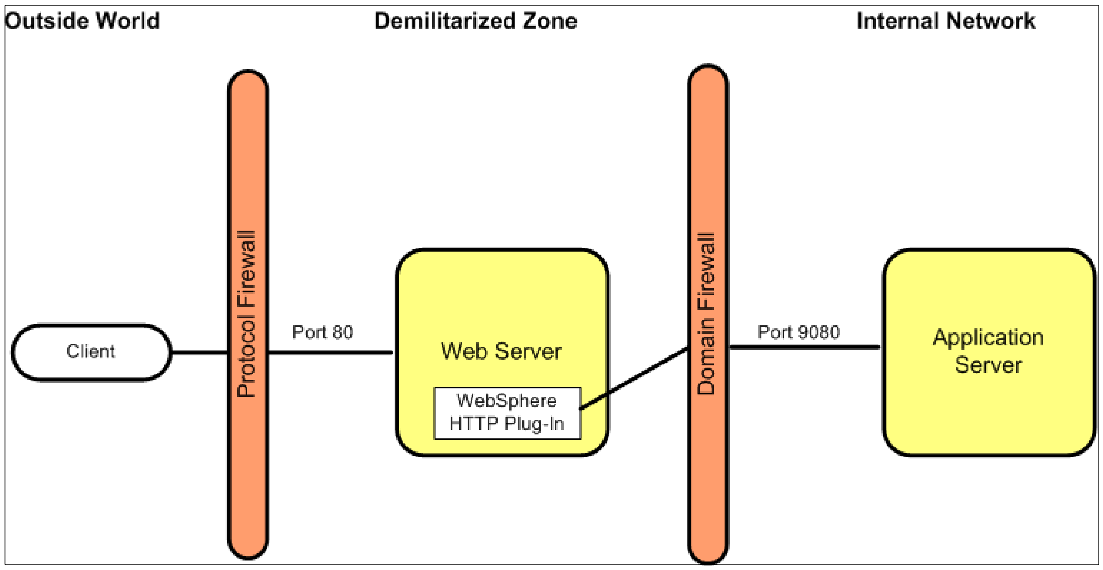


Figure 3-2 Simple WebSphere HTTP Plug-In Architecture Diagram

The following observations can be made about this figure:

- The Web server may be any Web server and platform combination that is officially supported by an appropriate WebSphere HTTP plug-in. At the time of writing, the HTTP plug-ins shipped with Lotus Domino are available for Microsoft IIS (Win32) and IBM HTTP Server (AIX). Support for additional Web servers on a variety of platforms has been announced.
- HTTP (or HTTPS, not represented in this diagram) may be redirected to the application server.
- The application server may be either WebSphere Application Server V4, V5 or Lotus Domino R5 or Lotus Domino 6; it must simply respond to HTTP/S requests forwarded from the plug-in.
- While port 80 traffic is accepted into the Web server, the plug-in redirects appropriate traffic to the application server through port 9080. Note that the ports that are used when redirecting Web traffic are also completely configurable.
- The rules that define the traffic to be redirected to an application server, versus handled locally by the Web server, are configured in the plug-in's XML-based configuration file on the Web server.
- Multiple rules may be defined, routing requests to several back-end application servers based simply on the URL request.

Tip: More information about the WebSphere HTTP plug-in architecture can be found on the WebSphere InfoCenter site at:

`http://publib7b.boulder.ibm.com/wasinfo1/en/info/aes/ae/crun_plugins.html`

The past three significant releases of WebSphere Application Server have included slightly different technologies that could be used to interface an HTTP Server, such as Lotus Domino's HTTP Server, with WebSphere Application Server. Table 3-1 identifies the last three significant point releases of WebSphere Application Server, along with the mechanisms provided to interface WebSphere with an HTTP Server. For each milestone release, the matched version of Lotus Domino's HTTP server that was supported is included. Note that this table pertains to the Windows NT/2000 platform only; support on other platforms might be slightly different.

Table 3-1 Evolution of WebSphere-Domino connectivity options

	Supported Domino HTTP Versions	**Remote OSE**	**Servlet redirection**	**HTTP plug-in**
WebSphere Application Server 3.5 AE	Domino 5	Yes	Yes	No
WebSphere Application Server 4.x AE	Domino 5 and 6	No	No	Yes
WebSphere Application Server 5.x	Domino 6	No	No	Yes

The focus of this redbook is on WebSphere Application Server V5 and Lotus Domino 6 connectivity features.

Restriction: The redbook team used Domino version 6.0.1 and WebSphere Application Server 5.0 in the lab. Although some of the interface methods do already work, the configuration, Domino 6/WebSphere Application Server 5, is not yet supported. Look for the next maintenance releases of Domino 6 to see if the support for WebSphere Application Server 5.0 has been added.

Attention: If you are interested in connectivity options available in prior releases of WebSphere Application Server, refer to the prior release of this redbook, *Applying the Patterns for e-business to Domino and WebSphere Scenarios*, SG24-6255, or other redbooks associated with IBM WebSphere Application Server Version 3.x or Version 4.x.

Tip: As stated in the WebSphere Application Server InfoCenter, a simple HTTP server is included in the WebSphere Application Server installation that responds, by default, to port 9080. However, this HTTP server should never be used in a production environment! An external HTTP server, such as IBM HTTP Server or Lotus Domino's HTTP server, should always be used, and is represented as a separate logical node in each of the Hybrid Runtime patterns described later in this chapter.

3.2 Domino-WebSphere Hybrid Runtime patterns

Several Hybrid Runtime patterns may be utilized in a mixed Domino and WebSphere Application Server environment. There are two common application types that emerge when analyzing Application Integration patterns associated with these products.

The first is based on the premise that a pre-existing Domino application requires an interface to WebSphere Application Server to support J2EE functionality, or interfaces to other back-end systems through software components provided in the WebSphere Application Server environment. An application of this nature may be described as being *collaboration-centric*.

The second typical application type involves an existing J2EE-based application that lives in the WebSphere Application Server realm, requiring access to one or more back-end Domino and relational databases. An example of this scenario may include a J2EE application that presents the contents of a view in a Domino database. Applications architected in this fashion are typically associated with more robust transaction-based processing, or *transaction-centric* systems, which may include some collaborative components, such as real-time chat interfaces with customer support personnel.

While Lotus Domino on it's own may be utilized as a transaction-centric system, such as an e-commerce Web site, there are limiting factors in the product itself that might restrict its scalability and flexibility. One of these limiting factors is the lack of a robust transaction processing model in Lotus Domino itself. Typical transaction-centric systems must interface with one or more back-end systems to complete a given transaction initiated by an end-user. Enterprise Java Beans (EJBs) in J2EE systems inherently provide robust and scalable transaction processing mechanisms that manage concurrent data access and provide data consistency among multiple systems, even in the presence of failures. By itself, Lotus Domino does not include the robust and scalable transaction processing mechanisms that are required in medium to large transaction-centric Web sites.

Similarly, while components can be built onto the WebSphere Application Server to address the needs of a collaborative workspace, Lotus Domino and other Lotus Software products provide this capability out-of-the-box.

Our discussion of Domino-WebSphere Hybrid Runtime patterns includes patterns that are both collaboration-centric and transaction-centric. The following Domino-WebSphere Hybrid Runtime patterns are described:

- Lotus Domino application with WebSphere Application Server services: Single server
- Lotus Domino application with WebSphere Application Server services: Multiple servers

- Lotus Domino and WebSphere Application Server with Web redirector
- WebSphere Application Server application with Lotus Domino services
- Lotus Domino providing Web services with WebSphere Application Server

For each of these patterns, the following information is provided:

- Situations in which the described pattern should be utilized.
- The typical Business patterns that might apply to the described Hybrid Runtime pattern, such as Self-Service, Collaboration or Information Aggregation. In some cases, the extent of either application or access integration that is found in the pattern is described, which may include specific access integration technologies that enable Lotus Domino to work with WebSphere Application Server.
- Benefits and limitations found in the use of the pattern.

As mentioned previously, the patterns described in this section are Hybrid Runtime patterns, in part because of their tie to specific IBM software products. In some cases, however, diagrams in the following subsections contain generic node names rather than actual product names for the sake of simplicity, or to emphasize the fact that products are indeed interchangeable in specific logical nodes. In some cases, if a generic node name is provided, there may be multiple actual products available to serve the functionality required. Table 3-2 describes typical products that may map to generic node names presented on included diagrams, when generic node names are provided.

Table 3-2 Generic node names and typical representative products

Generic node name	Typical product
HTTP server/Web server	IBM HTTP Server, Lotus Domino HTTP Server, Microsoft Internet Information Server (IIS), or other Web servers supported by the WebSphere HTTP plug-in technology
Web application server	IBM WebSphere Application Server
Collaboration server	IBM Lotus Domino and/or IBM Lotus Sametime for real-time collaboration
Database	IBM DB2
Directory	IBM Directory Server, Lotus Domino
Firewall	IBM Tivoli Firewall, Cisco PIX Firewall

Both the firewall and directory server have been included in some Runtime pattern figures. This does not imply that every situation will require a firewall or a separate directory server. The Runtime pattern diagrams illustrate best practices if your project requires a firewall or a separate directory server.

Note: It is important to remember that the Lotus Domino server and the Domino databases are not easily separated. Both typically exist on the same machine, unless an external storage mechanism is utilized for database storage (external hard drive arrays). For this reason, the logical nodes presented in any diagrams in this chapter with either the title Lotus Domino Server or Collaboration Server also imply the existence of one or more Lotus Domino databases within the same node.

3.2.1 Domino app with WebSphere Application Server services: Single server

This Runtime pattern is easily deployed, and fairly common in situations where WebSphere Application Server is being implemented for the first time within a Lotus Domino-based organization. The capabilities of WebSphere Application Server might be required to provide some increased measure of scalability by replacing the native servlet engine found in Lotus Domino with the more scalable and robust engine available in WebSphere Application Server.

While this pattern enables the use of more advanced features in J2EE, including Enterprise JavaBeans (EJBs), in most cases the use of EJBs would point to a more advanced system architecture with higher levels of availability that would indicate moving Domino and WebSphere Application Server to different physical nodes.

User authentication in this pattern would be performed by the Lotus Domino Directory.

Note: At the time of this writing, the HTTP plug-in for the Lotus Domino 6 HTTP server was only available for use with Microsoft IIS and IBM HTTP Server (AIX). Other Web servers will likely to be supported in upcoming maintenance releases of Lotus Domino.

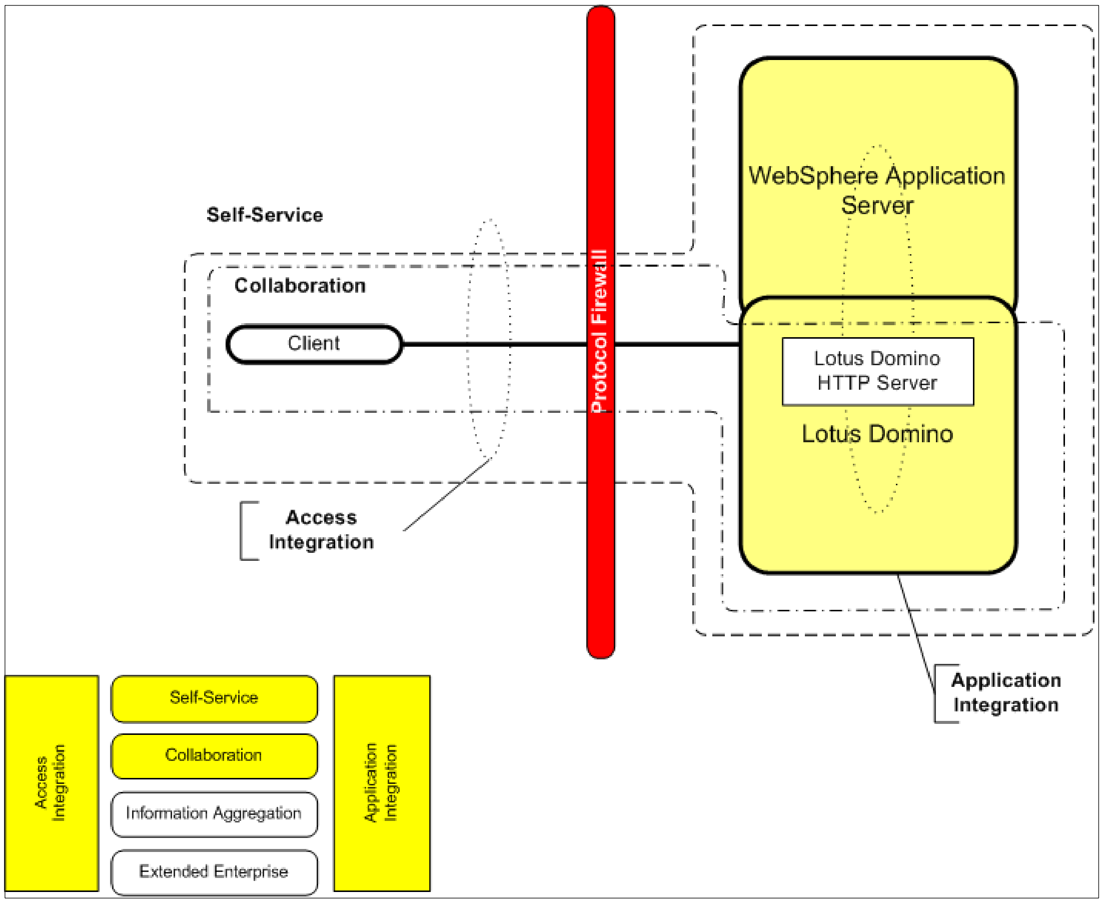


Figure 3-3 Domino application with WebSphere Application Server services: Single server

The most important nodes in the diagram in Figure 3-3 are explained as follows:

- The Client accesses the Lotus Domino HTTP server through a Protocol Firewall, which is intended to allow HTTP (port 80) access, but block all other ports.

- Lotus Domino and WebSphere Application Server are contained on a single physical server.
- Lotus Domino is serving as the user directory in this pattern, with no additional hardware components required.

Applicable Business patterns

The four patterns in action within this Hybrid Runtime pattern are:

- Self-Service

  This topology represents a pattern where users are interacting with a Web-based business system.

- Collaboration

  The collaborative functions of the Lotus Domino server are used for user-to-user interaction.

- Application Integration

  We can see application integration between Lotus Domino and WebSphere to provide integration between the two servers. In this topology, the integration would typically be done by accessing the other servers' classes directly through shared resources.

- Access Integration

  Access Integration can be seen between the HTTP task in Domino and both the Domino server engine and the WebSphere server. This access integration exists due to the HTTP task, which is used by both servers (Domino and WebSphere Application Server).

Lotus Domino and WebSphere Application Server integration with this Runtime pattern will usually be achieved by taking advantage of shared resources, for example, calling a Java bean directly from a Domino agent inside a Domino Database. Conversely, since the Domino database and WebSphere Application Server are contained within the same physical node, a Java bean could directly manipulate the contents of a Domino database in the Domino server through local Domino Objects for Java classes, instead of remote classes that require the use of IIOP.

If it is known, however, that the future of the application will include the physical separation of Lotus Domino from WebSphere Application Server, it is advised to use remote access (IIOP) with Domino for Java Objects to simplify the work required in breaking apart these application servers.

Guidelines for use

This Runtime pattern is appropriate in situations where the transactional, functional or scalability requirements are not enormous, and are easily contained within the bounds of a single server. In other words, this pattern is appropriate for collaboration-centric Web sites, especially in intranet applications, or in relatively simple, limited transaction-based systems that are built upon the Domino platform. This pattern might also be useful in test situations.

Examples of such sites might be:

- A company Web site where the majority of content resides within Domino databases, some selected functionality is provided by WebSphere, but only a relatively small number of users per day need that functionality.
- Web sites where Domino is the preferred deployment platform, without extensive requirements for transaction processing.

- Sites where the main focus is user collaboration, either with staff or with other users. For example, WebSphere may be used to control a host session via WebSphere Host Publisher.

In all of these example sites, the common thread is that WebSphere has a relatively minor part to play in the functionality of the site, providing ancillary services to a primary Domino application, such as servlet or JSP containment. This Runtime pattern is also well suited to test environments.

Benefits and limitations

The key benefits of this pattern include simple configuration and maintenance, as well as low hardware costs.

However, since there is no physical separation between Domino and WebSphere Application Server, this pattern does not scale well. It does provide a good starting point for implementations, however, if access interfaces are considered that allow for remote access versus those that rely exclusively on local access (such as local Domino Objects for Java).

3.2.2 Domino app with WebSphere Application Server services: Multiple servers

This Runtime pattern is a logical extension of Section 3.2.1, "Domino app with WebSphere Application Server services: Single server" on page 37. It presumes the use of a Domino server for all client intervention through Domino's own HTTP server, but separates the WebSphere Application Server onto a separate physical node, in this case behind the domain firewall, allowing for improved performance and scalability.

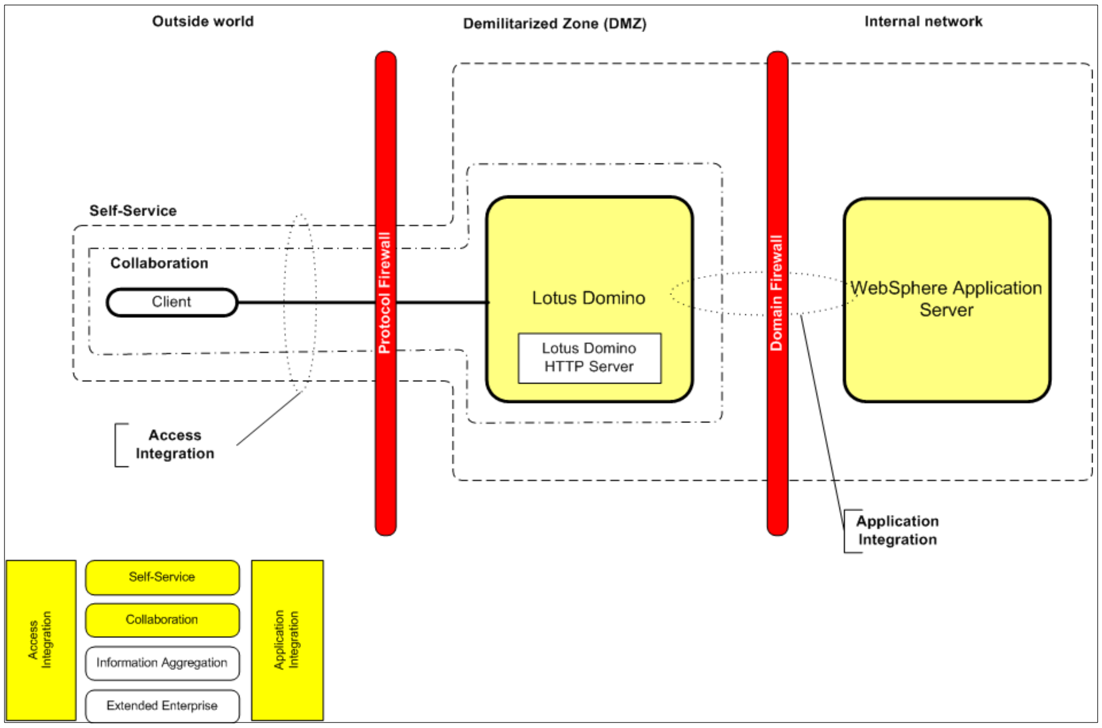


Figure 3-4 Domino application with WebSphere Application Server services: Multiple servers

The most important nodes in the diagram in Figure 3-4 are the following:

- The Client accesses the Lotus Domino HTTP server through a Protocol Firewall, which is intended to allow traffic through port 80 (HTTP) or 443 (HTTPS). All other ports are blocked.
- Lotus Domino's HTTP server is used exclusively to provide content to the client. The HTTP plug-in is utilized to redirect all servlet requests to the WebSphere Application Server.
- The Domain Firewall restricts access to the internal network to calls originating in the DMZ only; other traffic is blocked.
- The WebSphere Application Server is contained exclusively behind the Domain Firewall, providing an extra measure of security.
- Lotus Domino is providing all user directory services. WebSphere Application Server accesses the directory services through LDAP.

Applicable Business patterns

The patterns in action within this topology are:

- Self-Service

  The whole Runtime pattern represents a situation where users are interacting with a Web-based business system. This business system may consist of Domino components, but the addition of WebSphere Application Server opens up access to other enterprise-wide systems that may be accessed through EJBs in JSPs or servlets.

- Collaboration

  The collaborative functions of the Domino server are used for user-to-user interaction.

- Application Integration

  Domino and WebSphere are integrated solutions in this pattern. Application Integration between these products may be achieved through servlet redirection from Lotus Domino to WebSphere Application Server through the use of the IBM WebSphere HTTP plug-in.

- Access Integration

  Access Integration can be seen between the HTTP task in Domino and both the Domino server engine and the WebSphere Application Server. This access integration exists due to the HTTP task, which is responsible for the presentation of both servers (Domino and WebSphere Application Server).

There are a number of options for application integration between Domino and WebSphere; they are described in detail in 3.3, "WebSphere-Domino interface methods" on page 48.

From the perspective of Domino Server to WebSphere Application Server integration, the two products can be configured through the Domino Administrator, setting up the Domino HTTP task to utilize a third-party servlet engine (WebSphere Application Server) instead of it's own servlet engine.

Guidelines for use

This Runtime pattern is common in sites that provide more than just a single application or system to the Web users. It is also relevant when:

- Domino is the preferred application environment historically.
- The application requires more than trivial use of Domino's features, like Workflow, Collaboration, Mail and Calendar integration, and so on.

- Internal users create Web content from a Notes client which may have workflow associated with it.
- The site is collaboration-centric rather than transaction-centric.

In this Runtime pattern, the Domino server will authenticate users if required.

Benefits and limitations

The benefits of this Runtime pattern are:

- It scales well, both horizontally and vertically.
- It can provide increased security, since you are able to put WebSphere behind a firewall in the internal network rather than keep it within the DMZ.
- It is relatively simple to implement since it does not require special coding: it is configured, not coded.

When using Lotus Domino Release 5.x, one of the limitations of this Runtime pattern is that the only HTTP server that can used in the environment is the Domino HTTP task or MS IIS, but only on the same physical server. This serious limitation introduced horizontal scalability issues at high usage at the Web server logical node.

Lotus Domino Release 6.0 eliminates this restriction in two ways. First, the HTTP Server was re-written, and is much more scalable than prior releases. Second, a new Domino HTTP plug-In architecture was introduced that allows alternative HTTP servers to be used instead of the Domino HTTP server. This plug-in uses the same architecture as the HTTP plug-in for WebSphere Application Server. This mechanism allows the HTTP server to be physically separated from the Domino Application Server, and is further illustrated in the next section.

3.2.3 Web redirector with Domino and WebSphere Application Server

This Runtime pattern takes advantage of the WebSphere HTTP plug-in, which allows Lotus Domino and WebSphere Application Server to be physically separated from a Web server that is providing HTTP access to system information. This feature allows a Web server to exist by itself in the DMZ of a typical topology, and allows the Domino server to be placed behind the domain firewall, where other application servers, such as WebSphere Application Server, would typically reside.

This is the first pattern described that utilizes the WebSphere HTTP plug-in technology discussed at the beginning of this chapter, allowing the Web server to be physically separated from an application server.

User authentication using this pattern is provided through the Lotus Domino Directory.

Note: Lotus Domino includes robust security features that do a superb job of protecting Domino databases, even when they are placed in the DMZ and not behind a domain firewall, as long as Domino and all associated databases are configured properly through server security settings and database Access Control List (ACL) settings. While one benefit of the Domino HTTP plug-in architecture is in the elimination of Denial of Service (DOS) attacks directly on the Domino server, the primary benefits are in the areas of scalability and flexibility in the choice of HTTP servers utilized in an environment.

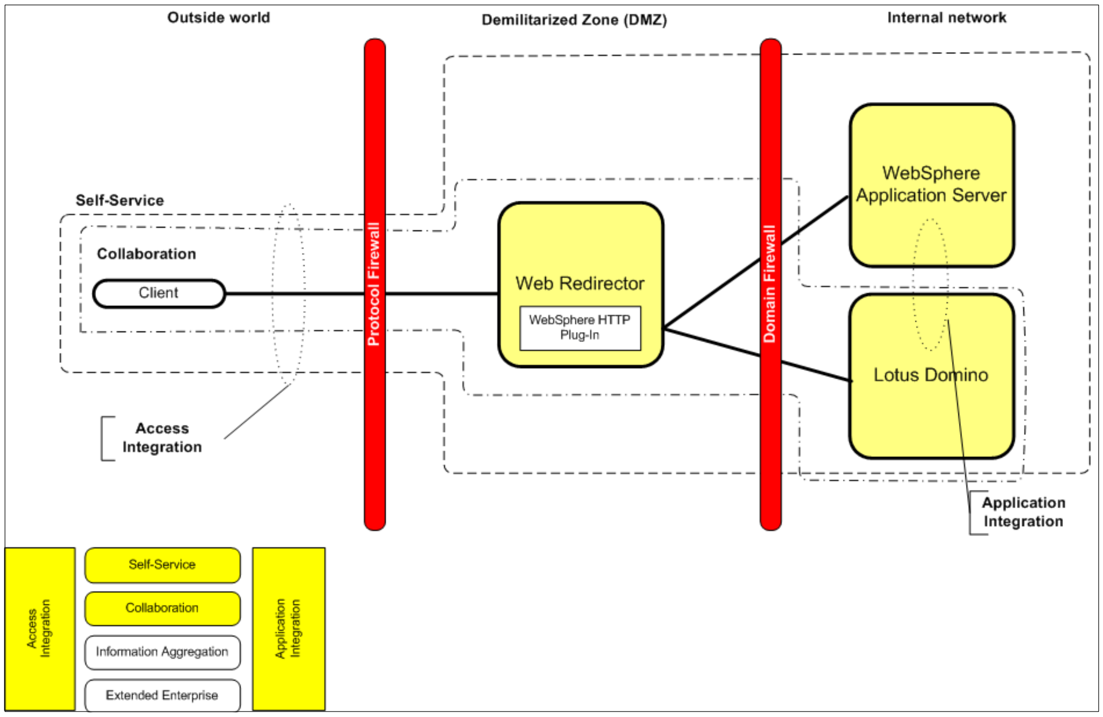


Figure 3-5 Domino and WebSphere with Web redirector

The most important nodes in Figure 3-5 are as follows:

- The Client accesses the Lotus Domino HTTP server through a Protocol Firewall, which is intended to allow only traffic through port 80 (HTTP) or 443 (HTTPS). All other ports are blocked.
- The only server in the DMZ is the Web server, which acts as a Web server *redirector*, directing incoming requests to either locally stored, static HTML pages, or to other Web servers based on the composition of the URL. The WebSphere HTTP plug-in is utilized on the Web server to perform this redirection task.
- The Domain Firewall restricts access to the internal network to calls originating in the DMZ; other traffic is blocked.
- The WebSphere Application Server is contained exclusively behind the Domain Firewall, providing an extra measure of security by allowing only traffic originating in the DMZ to reach the internal network.
- The Lotus Domino Server is contained behind the Domain Firewall, providing consistency with other application servers in this environment, such as WebSphere.

Applicable Business patterns

The patterns in action within this topology are:

- Self-Service

  In this pattern users interact with a Web-based business system. This business system may consist of Lotus Domino components, but the addition of WebSphere Application Server opens up access to other enterprise-wide systems that may be accessed through EJBs in JSPs or servlets. Potentially, Lotus Domino may also be utilized to access other enterprise-wide systems, through Lotus Domino technologies such as Lotus Enterprise Integrator™.

- Collaboration

  The collaborative functions of either Lotus Domino, or other Lotus or WebSphere Application Server collaborative products, apply to this pattern.

- Application Integration

  Lotus Domino and WebSphere are integrated in this pattern. The HTTP server (IBM HTTP Server, for instance) is also integrated through the Domino HTTP plug-in to work with both the WebSphere Application Server and Domino Application Server behind the domain firewall.

- Access Integration

  This is provided through the Web redirector, controlling access to one or more back-end systems through a common front-end server.

Guidelines for use

This pattern should be utilized in an environment that requires horizontal scalability at the HTTP server and application server level, in conjunction with protected data resources behind the domain firewall.

While this pattern specifies the use of the Lotus Domino Directory for user authentication, an alternative LDAP-based directory may also be introduced to handle this task, integrated with Lotus Domino, WebSphere Application Server and the Web server being used.

Benefits and limitations

The benefits of using this Runtime pattern include:

- A consistent architecture that places all persistent data nodes used in the system behind the domain firewall, including the Lotus Domino server and its associated databases.
- An elimination of Denial of Service attacks from the external network directly to the Lotus Domino server (by placing the Domino server behind the domain firewall).
- As additional HTTP plug-ins are provided with future releases of Lotus Domino and WebSphere Application Server, the ability to “plug in” additional Web servers will be provided.
- This plug-in architecture supports an added benefit of load-balancing. Requests can be redirected in a round-robin fashion, for example, to multiple, cloned application servers in a “group,” more evenly distributing load to multiple servers. Other patterns in this chapter do not include any form of load-balancing or fail-over mechanisms inherent in the pattern itself.

One limitation of this pattern is the availability of WebSphere HTTP plug-ins for specific Web server and operating system platform combinations, and the additional cost associated with purchasing and maintaining a separate physical Web server.

Tip: A complete explanation of the powerful WebSphere HTTP plug-in technology is beyond the scope of this redbook. For more detailed information, refer to the *IBM WebSphere V4.0 Advanced Edition Scalability and Availability,* SG24-6192, or the *IBM WebSphere Application Server V5.0 InfoCenter* on the Web, in the section “Web server plug-ins.”

3.2.4 WebSphere Application Server application with Domino services

Web sites that are primarily transaction-centric will typically utilize this pattern. Domino services may be provided to support a back-end calendaring system, electronic mail, or custom Domino databases, such as discussion groups or workflow-oriented databases.

Specifically, the pattern described here might be fairly rare, as a transaction-centric application utilizing WebSphere Application Server would also typically include a relational database component, such as IBM DB2, and not just Lotus Domino, which would provide only a subset of services. However, for the sake of simplification, and due to the focus on Domino-WebSphere integration, additional potential nodes have not been included.

From the perspective of user authentication, the most effective solution would be the inclusion of IBM Single Sign-On in this pattern, using either Lotus Domino's directory or an alternative, compatible LDAP solution, such as IBM Directory Server. For more information, see 3.4, "Utilizing IBM single sign-on" on page 55. For the sake of discussion, the Figure 3-6 includes a separate logical node for the Directory Server.

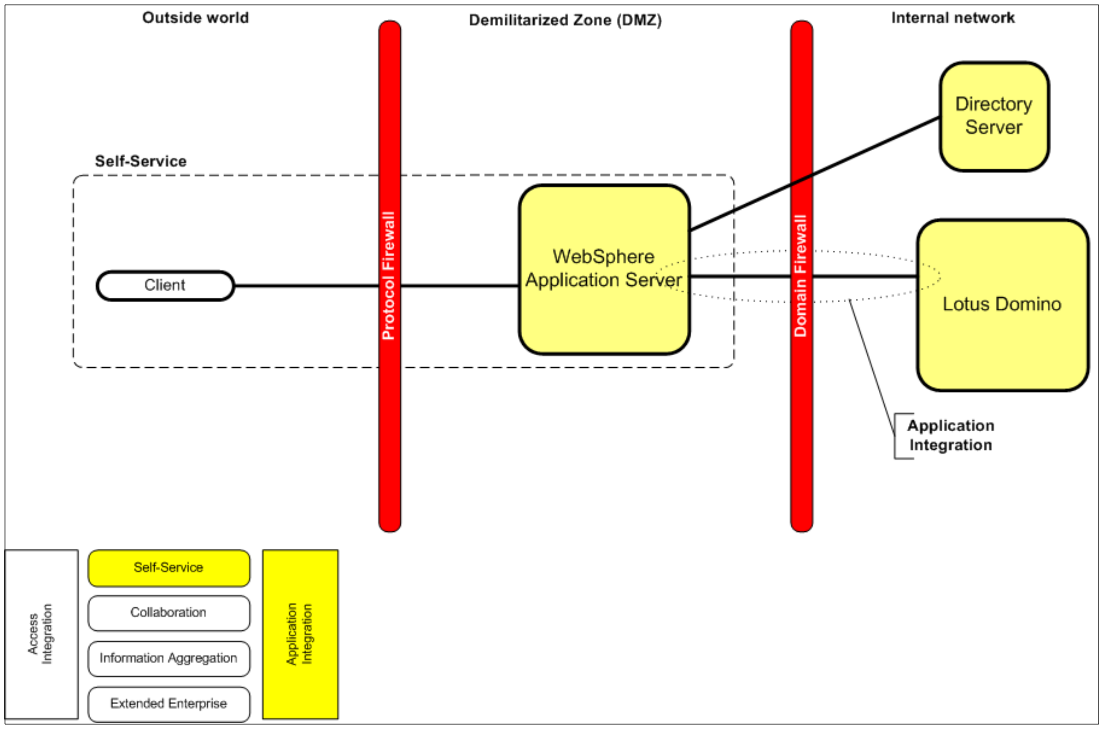


Figure 3-6 WebSphere Application Server with Domino services

The most important nodes in this diagram are explained as follows:

- The Client accesses the WebSphere Application Server through a Protocol Firewall, which allows access only through port 80 (HTTP) or 443 (HTTPS). All other ports are blocked.
- Components executing in WebSphere Application Server, including JSPs, JavaBeans, or EJBs, may access Lotus Domino database resources remotely, through either Domino for Java Objects (local or remote) or Domino Collaboration Objects.
- The Domain Firewall restricts access to the internal network to calls originating in the DMZ; other traffic is blocked.
- A Directory Server is introduced, protected behind the Domain Firewall, to provide user authentication. This server may be Lotus Domino, or an alternative directory server, such as IBM Directory Server.

Applicable Business patterns

The patterns that form this Runtime pattern may include:

- Self-Service

  WebSphere Application Server is providing the entire user interface to the business in this Runtime pattern.

- Application Integration

  Lotus Domino and WebSphere Application Server are integrated through one or more of the interfacing mechanisms described in 3.3, “WebSphere-Domino interface methods” on page 48. Most typically the Domino Objects for Java interface method will be utilized to access Domino from WebSphere Application Server components when this pattern is used.

Guidelines for use

This Hybrid Runtime pattern is best used in traditional transaction-centric sites that may, out of necessity, include one or more collaborative components.

There are a number of methods that can be used to establish connections between WebSphere Application Server and Lotus Domino in this Runtime pattern, as described in the next section.

Benefits and limitations

The key benefits of this Runtime pattern are:

- Security

  It enables the Domino server to reside behind the firewall in the internal network with the other data sources. In that way, all of the sensitive data is stored internally, not in the DMZ.

- Scalability

  WebSphere is able to provide a consistent view and speed because it can be scaled to improve performance and fault-tolerance as required by incorporating the WebSphere Edge Server.

One of the limitations of this pattern is the lack of a dedicated HTTP server in the DMZ.

3.2.5 WebSphere Web services with Domino

This scenario depicts a Hybrid Runtime pattern that will likely become more prevalent as Web services mature and their use becomes more widespread.

This pattern assumes the use of WebSphere Application Server as a provider of Web services. Due to the nature of Web services, however, it is easy to imagine a Lotus Domino server providing this same functionality, simply swapping Lotus Domino and WebSphere Application Server in Figure 3-7. The Web service Provider, in this pattern, also serves the purpose of information aggregator, pulling information from Lotus Domino databases in conjunction with IBM DB2 relational databases through one or more published Web services.

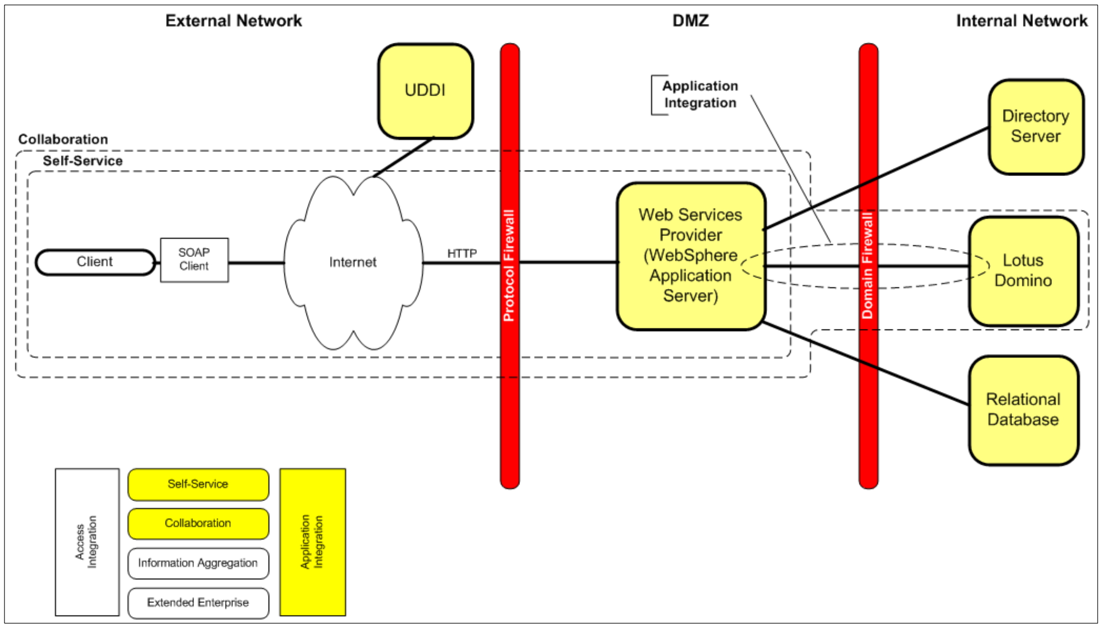


Figure 3-7 WebSphere Application Server providing Web services

Important notes about this figure:

- The Client node in this figure is the Web service Consumer, accessing the provider through the Internet.
- The Web service Provider, in this pattern example, is IBM WebSphere Application Server.
- The service provider has published information about available Web services to a UDDI Directory.
- This implementation of Web services uses HTTP as the transport protocol.
- The SOAP client represents additional functionality required to encode and decode SOAP messages used in Web services.

Applicable Business patterns

As indicated in Figure 3-7, the Business patterns that are applicable to this Runtime pattern are:

- Self-Service

  Web services may be deployed that allow end-users to reach into back-end business systems.

- Collaboration

  The Web services provided by the Lotus Domino server may center around collaborative capabilities found natively in Lotus Domino, including workflow processes, shared calendaring and scheduling, and messaging, that may be triggered or controlled by external events through a Web service interface.

- Application Integration

  Lotus Domino and WebSphere Application Server work together in this pattern. WebSphere Application Server integrates with Lotus Domino, as well as, potentially, other data stores.

Guidelines for use

This pattern may be the best choice in environments in which:

- Multiple application server platforms exist (including Microsoft .Net, J2EE platforms such as IBM WebSphere Application Server and Lotus Domino), in which the single common interoperability layer may be the use of standard, published Web services.
- Supply chain integration is required with other vendors through the Internet (extended enterprise).
- Internet-based clients (end users) require access to information through a standard Web services interface.

Benefits and limitations

The key benefit of this Runtime pattern is the use of standards-based Web services, allowing platform and implementation independence and interoperability between disparate systems. In the pattern described, a Web service is provided directly from the WebSphere Application Server. However, this same service may be expanded to include extended enterprise access (supply chain or business partner integration).

While there are many benefits to the use of Web services, in some situations, Web services may be less than advantageous for performance or other reasons. For example:

- In many manufacturing environments, Electronic Data Interchange (EDI) is already used by suppliers to communicate information from one organization to another. Adding a Web services layer to a legacy EDI system would require the added complexity of serializing, then deserializing, each EDI transaction simply for the sake of utilizing Web services. This process would result in performance inefficiencies and added complexity.
- Binary data, such as images (JPEG, for instance), would also require the same serialization and deserialization of messages. Images are typically already highly optimized and compressed for transmission; encapsulating binary information into XML would be slow and inefficient. Three potential, emerging standards in dealing with binary data in XML include *SOAP Messages with Attachments* (SwA), *Direct Internet Message Encapsulation* (DIME) and *Blocks Extensible Exchange Protocol* (BEEP).
- Some existing legacy systems may not be Web services-compliant, and may never be.

Due to the relative immaturity of Web services, another potential limitation of this pattern lies in the lack of widely supported and standardized security mechanisms tied to Web services. IBM, Microsoft, VeriSign and other organizations are attempting to ratify security mechanisms, known as Web Services Security (WS-Security). For more information related to this new standards effort, WS-Security, refer to:

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

As Web services and related standards evolve, best practices to handle many of these shortcomings will likely emerge.

3.2.6 Summary of Domino-WebSphere Hybrid Runtime patterns

Table 3-3 provides a summary of the significant benefits, limitations, and typical usage guidelines for the patterns described in this chapter.

Table 3-3 Domino-WebSphere Hybrid Runtime Patterns Comparison

	Implementation effort	Scalability	Security
Domino-WebSphere Single Server	Low	Low	Low
Domino-WebSphere Multiple Server	Low-Medium	Low-Medium	Low
WebSphere Application with Domino Services	Medium	Medium-High	High
Domino and WebSphere with Shared Web Redirector	Medium	High	High
Domino and WebSphere Web Services	High	High	Still under development

3.3 WebSphere-Domino interface methods

In this section, we introduce the various methods that are available to the applications developer when accessing information stored in a Lotus Domino database from a WebSphere Application Server node. Each option has distinct advantages and disadvantages.

These interface methods differ from the connectivity methods described previously in this chapter in that these methods focus specifically on access to Lotus Domino databases from servlets, JSPs, JavaBeans, or EJBs that might exist in a WebSphere Application server application. These interface methods do not assume that Lotus Domino is in any way configured to handle service requests directly from end-users, but instead assumes that WebSphere Application Server (or WebSphere ancillary products, such as WebSphere Portal Server) will service all direct end-user requests. All of the Hybrid Runtime patterns described previously may be used in conjunction with these interface methods.

The interface methods discussed in this section are:

- Domino Objects for Java: Local access
- Domino Objects for Java: Remote access (CORBA/IIOP)
- Domino Collaboration Objects (DCO)
- Web Services
- Domino Tag Libraries (JSP)
- Java Database Connectivity (JDBC)

After the discussion of each of these integration methods, a table that compares them methods in summary form is provided.

Note: Since the emphasis of this redbook is on high-level patterns that apply to Domino-WebSphere integration, the actual integration methods between the products discussed are not as detailed as they might be in other Redbooks. For a more detailed explanation of integration methods, including examples, refer to *Domino and WebSphere Together,* SG24-5955.

3.3.1 Domino Objects for Java: Local access

This method of interfacing WebSphere applications to Domino databases accommodates the highest level of interfacing possible directly from Java, without resorting to custom Java wrappers around the underlying Notes API through the Java Native Interface (JNI).

Typical usage scenarios

Local Domino Objects for Java may be utilized when a component of a WebSphere application must access Domino data on either a local or remote system. Typically, the local access versions of these classes are utilized in smaller environments, such as the pattern described in 3.2.1, "Domino app with WebSphere Application Server services: Single server" on page 37.

This interfacing method is usually not used on large-scale systems.

Benefits and limitations

In order for local Java access to work, a copy of the Lotus Notes Client must be installed on the same system that is utilized by the Domino Objects for Java system. Since the local Notes Client API is utilized by this Domino for Java implementation, either the database being accessed must be contained on the same physical node as the entity accessing it, or port 1352 (standard Lotus Notes port) must be open between the executing physical node and the server with the Domino database. In some environments, the opening of this port for communications may present a security issue based on existing corporate standards.

When utilizing local Domino resources, a developer must be aware of thread initialization and termination of Domino sessions, which may be cumbersome in the long-run, especially when developing anything more than a fairly simple J2EE body of work (EJB, servlet, or JSP). Improperly terminating threads that have been initialized will result in system crashes.

3.3.2 Domino Objects for Java: Remote access (CORBA/IIOP)

Like local access Domino Objects for Java, this interfacing mechanism allows for the highest level of interaction possible through Java. The difference between these mechanisms is that sessions created with this flavor of Java objects utilize Common Object Request Broker Architecture (CORBA). The Internet Inter-Orb Protocol (IIOP) is utilized as the transport protocol of CORBA requests to and from the Domino server; hence, the DIIOP task must be online and functioning on the Domino server to use Remote access. The default port that IIOP uses on the Lotus Domino server is 63148.

Typical usage scenarios

This interfacing method should be used when maximum flexibility is required not only in programmatically accessing Domino databases from J2EE application components, but also when flexibility in network topology is required.

Benefits and limitations

The primary benefits of this interface method include maximum flexibility in accessing Lotus Domino databases through the rich set of Domino Objects for Java that are available, and the ability to physically separate the WebSphere Application Server from the Lotus Domino server. In addition, the developer using remote Java calls does not have to worry about the thread initialization and termination code described in the previous section, making initial development and ongoing maintenance a much simpler prospect.

One of the limitations of this mechanism may lie in various scalability issues that have been reported in the Lotus Domino IIOP task. These issues, primarily related to session creation and appropriate time-out issues, should be resolved in Lotus Domino 6, but could not be fully investigated at the time of writing.

Tip: For a more detailed presentation of the advantages and disadvantages of local versus remote Java calls, see *Domino Integration: Tips for Working with Domino Objects* in WebSphere Advisor Magazine, January 2001 edition.

3.3.3 Domino Collaboration Objects (DCO)

Domino Collaboration Objects (DCO) provide an alternative mechanism to the application developer interested in accessing core Domino system capabilities without requiring full knowledge of the Domino back-end Java classes. The DCOs are provided with the Lotus Domino Toolkit for Java/CORBA, Release 5.0.8.

It should also be noted that “under the covers” the DCOs utilize the remote (CORBA) flavor of Domino Objects for Java, so some of the same benefits and limitations apply.

Typical usage scenarios

Unlike Local or Remote (CORBA) Java access to Domino data, DCOs provide more specific functionality in the form of Java beans that provide the following capabilities (in the 5.0.8 release of the Domino for Java Toolkit):

- User login and session authentication
- Sending an electronic mail
- Working with calendar entries

DCOs can be used from within Java applications, Java applets, or servlets (including JSPs).

Benefits and limitations

The primary benefits of utilizing DCO versus the other Domino access methods via the Java back-end classes are:

- DCO is focused on specific functionality provided via Domino (sending e-mail or creating calendar entries).
- The developer is not required to understand the underlying Domino Object Model.
- The developer is not required to fully understand the structure of specific Domino databases, such as those based on the standard mail template, to perform functions provided by the DCO.

The clear limitation found in the use of DCOs is that only functionality provided through published DCO Java beans is available. Any functional requirements that are outside the scope of the DCO will require the developer to revert to the use of the full Domino object model.

Attention: At the time of this writing, we could not confirm that the Domino Collaboration Objects as provided in the 5.0.8 release of the Domino for Java Toolkit worked as advertised when using Lotus Domino 6.

3.3.4 Web services

Unlike the other interface methods described in this section, the use of Web services applies to not only IBM-specific solutions, but also opens up real interoperability between disparate platforms, including Microsoft .Net, Sun Open Net Environment (ONE) and a variety of open-source solutions. Any vendor that adheres to the W3C standards and recommendations related to Web services can serve as a provider or consumer of Web services.

The primary technologies utilized by both consumers and providers of Web services include:

- eXtensible Markup Language (XML) - A self-describing, extensible format for encoding data.
- Simple Object Access Protocol (SOAP) - A lightweight protocol for exchange of information in a decentralized environment, based on XML. HTTP servers are traditionally utilized as the transport mechanism for SOAP information. However, in theory, SOAP may utilize other transport mechanisms.
- Universal Description, Discovery and Integration (UDDI) is a vendor-neutral, standard directory that contains interface information about registered Web service providers.
- Web Services Description Language (WSDL) is a W3C sanctioned XML format used to describe the nature of published Web services.

The advantages of Web services in any integration framework are many, including:

- Web services are self-contained and abstracted from the implementation layer. Web services deployed on the IBM J2EE WebSphere Application Server platform, for example, will interoperate with Web services deployed on a Microsoft .Net platform.
- Since Web services are deployed using XML, messages sent to and from a Web services server are self-describing and can be easily parsed in any language with readily available code libraries.
- Web services can encapsulate existing business objects, including EJBs, JavaBeans, or DCOM objects, and tools are widespread that easily encapsulate business object entities in a Web services wrapper.

For the sake of discussion in this redbook, we are interested in providing a Web service, hosted in WebSphere Application Server, that encapsulates Domino-specific functionality. Alternatively, Lotus Domino may serve as a Web service provider. However, as just described, there is nothing inherent in this interface method that would prevent a non-WebSphere or non-Domino application from achieving the same general interface. Hence the beauty of Web services: relatively simple interoperability between dissimilar and potentially geographically dispersed systems.

Typical usage scenarios

Web services can be implemented in a number of ways. From either the Domino or WebSphere perspective, Web services can be utilized or published.

Some examples of utilizing Web services to implement interfaces between Domino or WebSphere Application Server are:

- A Domino server may provide a Web service that provides user information contained in the Domino directory, such as an individual's address or phone number.
- An agent written in Domino may act as the consumer as a Web service contained on a WebSphere server to obtain stock price information stored in an IBM DB2 database.
- A WebSphere server may act as the provider for a Web service that aggregates data from a Domino database and an Enterprise Resource Planning (ERP) system. The Web

service may provide information, such as an employee's current health care benefits selection and their current mailing address from the Domino Directory.

In Lotus Domino, Web services can be written using LotusScript or Java-based agents. In LotusScript agents, a SOAP-encoded message can be manually assembled or disassembled through code, or more simply through the use of the Microsoft Toolkit for SOAP, which includes an MSSOAP COM object. In Java agents, the classes provided in Apache SOAP and Apache XERCES projects (both freely available on `http://www.apache.org`) provide similar capabilities, simplifying the steps required to create and decipher SOAP messages.

When constructing a Web service that will be served using WebSphere Application Server, Apache SOAP and Apache XERCES projects should be utilized to assist in the creation of either Web service provider or consumer services.

Benefits and limitations

There are several benefits in the use of this interfacing mechanism, such as:

- Web services are self-describing and fairly simple to parse using standard toolkits since they are based on XML.
- Web services are based on open standards, and will interoperate well with Web services provided on a variety of platforms, regardless of their underlying implementation.
- Web services may serve as data aggregators, providing read and write access to not only Lotus Domino databases, but also relational databases, such as DB2.

One of the limitations of Web services is the added complexity of implementation. For example, a Web service provided in WebSphere Application Server that accesses a simple document in a Lotus Domino database must not only use the Domino Objects for Java to access the database, but must also use SOAP to encode and decode the service envelope and header, (potentially) publish the Web service in a UDDI directory, and create a properly formed WSDL XML body that accurately describes the service provided.

Fortunately, modern integrated development environments (IDEs), such as IBM WebSphere Studio Application Developer, include various wizards that simplify this process dramatically.

3.3.5 Domino tag libraries (JSP)

The Domino tag libraries provide an alternative mechanism for accessing Lotus Domino databases without the need to learn Domino Objects for Java or DCOs. These tag libraries provide high-level access to Domino forms, views, and other Domino design elements. Like DCOs, it should be noted that the Domino tag libraries may utilize the remote (CORBA) implementation of Domino Objects for Java, or the local implementation of this mechanism for single-server patterns.

JSPs, of course, are served from a WebSphere Application Server environment, providing a significant amount of scalability, as well as integration with other systems.

Typical usage scenarios

Domino tag libraries are utilized in JSP pages that require access to Lotus Domino database information. The Domino tag libraries may be the best choice for programmatically accessing Lotus Domino data from JSP pages for developers accustomed to JSP page development and the use of JSP tag libraries. Single sign-on is supported through the use of the Domino tag libraries, in addition to the ability to execute full-text searches and run server agents.

In addition, several tags are available that allow state and environment information to be used within a JSP page. These tags include, for instance:

- The ability to determine the ACL settings for the current user
- The ability to create or delete documents in a database
- The ability to execute server-side Domino agents
- The ability to execute @Formula commands

Benefits and limitations

The most obvious benefit of the Domino tag libraries is that developers do not need an understanding of the Domino Objects for Java. Instead, developers need to understand the Domino tag libraries and the use of JSP. Therefore, this interfacing method is most suited to teams with servlet and JSP development experience, but little direct Domino Objects for Java knowledge.

Another benefit of Domino tag libraries is in the level of integration to be provided in IBM WebSphere Studio Application Developer. Lotus will soon be releasing a plug-in for the WebSphere Studio Application Developer environment that provides more direct integration of JSP development in WebSphere Studio Application Developer with existing Lotus Domino databases. For example, a form in a Lotus Domino database may be “imported” into a WebSphere Studio Application Developer project as a JSP page utilizing Domino tag libraries to represent various design elements from the original Domino form.

From another perspective, since JSPs would be served from a WebSphere Application Server, they provide a wide range of scalability options.

The disadvantage of using Domino tag libraries is that functionality is limited to what is provided in the JSP tags themselves. An analysis of the functional requirements of an application, or a specific JSP page, must take into account the specific tags available in the Domino tag libraries to ensure that functional requirements can be met.

Tip: Despite this disadvantage, another considerable benefit tied to the use of Domino tag libraries is the introduction of more generic, mainstream J2EE development, which is a core foundation of the Lotus NextGen strategy. The use of Domino tag libraries, then, may be considered a stepping stone for Lotus Domino developers into the J2EE world.

Lotus Domino Toolkit for WebSphere Studio

The Lotus Domino Toolkit for WebSphere Studio is a product that lets application developers access Lotus Domino applications and data from WebSphere Studio. The toolkit is built on the Eclipse open source platform and it shares a common look and feel with other WebSphere toolkits. With the toolkit, it is possible to see deployed Domino applications on the Domino server and reuse Domino forms, views, and agents in J2EE applications.

The toolkit isn't a programmable method to access Domino data itself. It generates Domino custom tag code automatically for Domino design elements when dragging those elements to a Java Server Page in WebSphere Studio. The toolkit installs the Domino custom JSP tags for the WebSphere Studio environment for easy use and for access to help and samples.

This toolkit helps Notes/Domino application developers understand and implement the new Domino custom JSP tags that ship with Domino 6. Developers can use their existing development skills and blend Domino's application platform advantages for collaboration with the transactional aspects of WebSphere applications designed for the J2EE architecture.

For J2EE developers developing with WebSphere Studio, the toolkit allows access to the Notes/Domino data store and application features within the context of the J2EE technology. The toolkit understands how to connect and what's needed to connect to Domino data, and it configures JSP tags with attributes automatically. Without the toolkit, the developer needs to understand how to connect to Domino data, set up the connection, and code the JSP tags manually.

The Lotus Domino Toolkit for WebSphere Studio will be included in a future version of Domino Designer 6. It requires both Domino 6 and WebSphere Studio V5. Final code is currently scheduled to be generally available with Lotus Domino Designer in Q2 2003.

3.3.6 Lotus Domino Driver for JDBC (LDDJ)

It is possible to access data stored in a Lotus Domino database through standard Java JDBC calls. Lotus provides a Level 2 JDBC driver (Lotus Domino Driver for JDBC, or LDDJ) that makes a Lotus Domino database look like a true relational database.

Typical usage scenarios

JDBC access to Lotus Domino data may be used in situations where the application development team is not familiar with the Domino Object Model, and has no need to perform any tasks beyond simple data access (read/write).

Benefits and limitations

The primary benefit of the LDDJ is that those with little to no experience with the Domino Object Model can access data contained in a Domino database using a familiar JDBC mechanism.

Some of the limitations inherent in the implementation of LDDJ are:

- LDDJ can be utilized only on the Windows/32 platform (NT/2K).
- The LDDJ driver requires the existence of Lotus Notes or Lotus Domino on the same system executing Lotus Domino JDBC calls, requiring additional effort during system installation and ongoing maintenance, compared to remote (CORBA) Java access, which requires only a “skeleton” framework on the accessing system.
- Lotus Domino is not a relational database, even though LDDJ makes it look like one. Overly complex SQL statements that attempt to translate a Lotus Domino database into a relational database may be the death-knell of any application that is expected to appropriately and predictably scale up to a moderate to large system.
- The JDBC driver is not a full implementation of JDBC, and does not support all SQL statements.

Important: Do not use the LDDJ in an effort to turn a Lotus Domino database into a relational database. Doing so will yield a system that cannot scale. If highly relational data is required in a system, use a true relational database system, such as IBM DB2.

3.3.7 Summary of Domino-WebSphere integration methods

Table 3-4 summarizes the integration methods between Domino and WebSphere discussed in this chapter. Note that scalability, an important factor to consider when comparing interfacing methods, is not included in this table. While the interfacing method should be considered, scalability is much more a function of the Hybrid Runtime pattern selected.

Table 3-4 Summary of Domino-WebSphere integration methods

	Domino Objects for Java: Local	Domino Objects for Java: Remote	DCO	Web services	Domino Tag Library (JSP)	LDDJ (JDBC)
Implementation Effort	Moderate	Moderate	Simple	Moderate to Difficult (Depends on implementation)	Moderate	Simple
Level of Integration Options and Flexibility	High	High	Low	High	Medium	Medium
Security Considerations	Default Notes ID on physical node running is used	Uses HTTP user ID and password via IIOP; may use SSO	Uses HTTP user ID and password via IIOP; may use SSO	Depends on implementation; may use Local or Remote Domino Objects for Java	Uses HTTP user ID and password via IIOP; may use SSO	The default Notes ID on the physical node running JDBC is used
Special Domino Considerations	None	Requires IIOP task	Requires IIOP task	Depends on implementation	Requires IIOP task	Win32 only; Lotus Notes DLLs required on executing physical node
Special WebSphere Considerations	Lotus Client or Server DLLs must be on WebSphere Server	None	None	Depends on implementation	None	Lotus Client or Server DLLs must be on WebSphere Server
Other	None	None	None	None	None	Transaction roll-backs not supported (immediate commits)

3.4 Utilizing IBM single sign-on

Single sign-on, or SSO, typically means different things to different people. In the scope of this redbook, we describe SSO in IBM product-specific terms. SSO is defined as the mechanism that shares authentication information between application servers, in our case, Lotus Domino and WebSphere Application Server. When SSO is enabled, a user who has been authenticated once by an application server will be automatically authenticated on other application servers in the same DNS domain.

The mechanism utilized by IBM's SSO technology requires the use of a token which is stored as a cookie in the user's browser. This token, referred to as a *Light-Weight Third Party Authentication* (LTPA) Token, contains data that uniquely identifies the user, such as the user's ID and a digital signature used to authenticate the token by the application server.

Special notes related to IBM SSO include:

- All application servers that take advantage of SSO must reside in the same DNS domain.

- All application servers must share the same user registry (directory). Supported directories include Lotus Domino (configured as an LDAP directory) and IBM Directory Server.
- Browsers accessing application servers must be configured to accept cookies, which are used to store a token containing authentication information.
- Optionally, SSO may be set up to work only on encrypted HTTPS connections (SSL). Passing the LTPA token over a non-encrypted connection would present a security vulnerability. If system security is a priority all connections should utilize SSL.
- While this IBM SSO solution will work with WebSphere and Domino, it is an IBM-specific solution that does not work with other application server vendors.

Tip: For a more detailed explanation of IBM single sign-on, refer to *IBM WebSphere V5.0 Security WebSphere Handbook Series,* SG24-6573.

The use of SSO applies to any one of the Domino-WebSphere Hybrid Runtime patterns described in this chapter, provided, of course, the products used are IBM SSO-compliant. For instance, the Directory Server, where indicated in patterns diagrams, can be Lotus Domino or IBM Directory Server.

3.5 Summary

In this chapter we have discussed the fundamentals of Lotus Domino and WebSphere Application Server interoperability, and the Hybrid Runtime patterns that are typically associated with the use of these products. In our analysis of each Hybrid Runtime pattern, we have enumerated the Business patterns that apply, as well as general guidelines for use, limitations and benefits. The next chapter will extend upon these fundamental patterns, introducing additional products to one or more of the patterns described in this chapter.

4

Integration: Further steps

In this chapter we extend the Hybrid Runtime patterns from the previous chapter to include other technologies commonly deployed with Domino and WebSphere.

These include patterns covering the various directory deployment and integration options available, including enabling single sign-on across many different products. Support for real-time collaboration using Lotus Sametime is also discussed.

4.1 Directory Integration patterns

Enterprises typically have many different directories, meaning administrators have difficulty maintaining them and end users need to remember multiple logins. As a result, many organizations are looking to consolidate a variety of directories and sources of people information into one Enterprise directory. This effort is greatly aided by the widespread support for the Lightweight Directory Access Protocol (LDAP) standard by the software industry.

A further requirement is to then allow single sign-on, where a user logs on to one application and is then authenticated against many other applications that trust the initial logon. While these requirements appear to be similar, they are actually quite different. Both are discussed in this section.

A common source for people-related information is directories from applications with messaging and collaboration features, the most popular of which are Lotus Domino and Microsoft Exchange.

Lotus Domino's directory has been LDAP-compatible since release 5, and provides several features for consolidating and authenticating against other Domino and LDAP directories.

Microsoft Exchange relies on the Windows NT4 domain in Exchange 5.5 and Active Directory on Windows 2000 for Exchange 2000. Active Directory is LDAP-compatible.

The many permutations of directory-related patterns with Domino 6 and WebSphere 5 can not be covered in this book, so the most common ones have been selected. These are:

- Using an external LDAP directory for user information and authentication
- Using both Active Directory and Domino Directory
- Using both an external LDAP directory and Domino Directory
- Using a security server to manage authentication and connections

4.1.1 Using an external directory for user information and authentication

Many organizations do not want the administrative overhead of maintaining more than one directory for user information and authentication. Using an external directory for these tasks is a way to get around this issue; it is illustrated in Figure 4-1 on page 59.

This pattern is a simple solution to this problem, one that doesn't require third party software (except the directory itself) and is easy to set up. However, it is only applicable to specific scenarios.

Note that all the patterns here assume that Domino is using the HTTP plug-in architecture discussed in the previous chapter. This provides the best security by avoiding having any data in the DMZ.

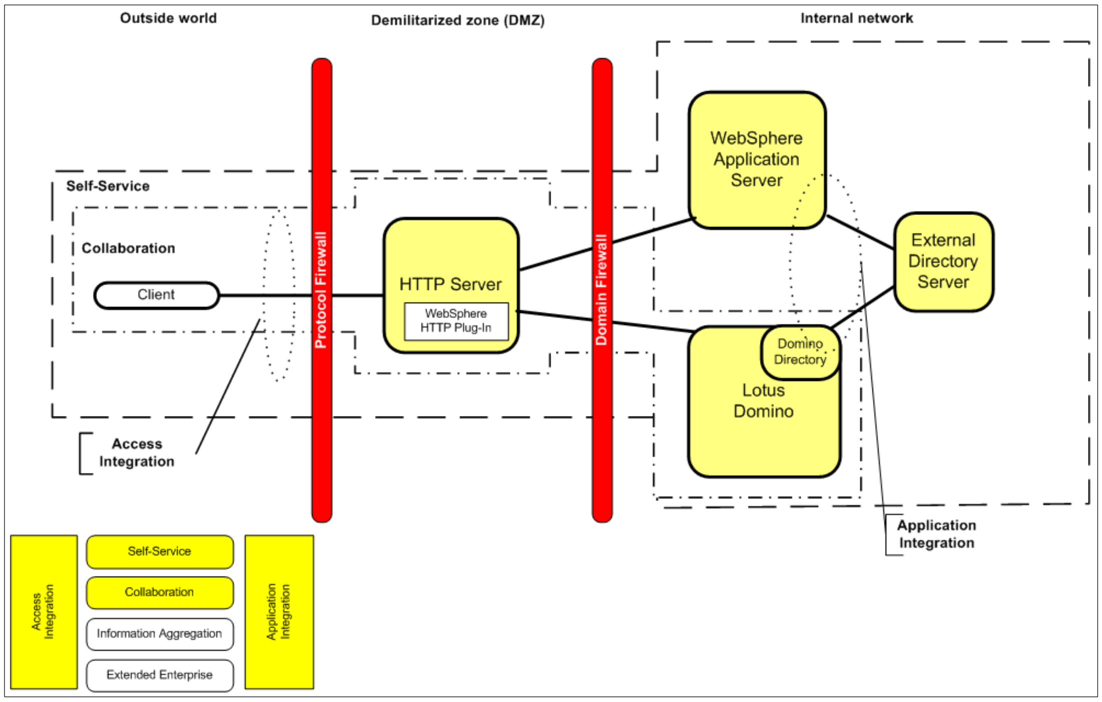


Figure 4-1 Domino and WebSphere Application Server using an external directory.

Applicable Business patterns

The four patterns which form this topology are:

- Self-Service

  One of the user's interactions with the environment is logically with WebSphere applications. That interaction represents the Self-Service pattern.

- Collaboration

  One of the user's interactions is logically with Domino collaborative applications. That interaction represents the Collaboration pattern.

- Access Integration

  Clearly, when both application servers are sharing a directory for authentication purposes, the directory is providing security integration which is a subset of access integration.

- Application Integration

  The directory, as well as acting as an access integrator, allows both Domino and WebSphere Application server to use the directory as storage for people- or resource-related information, and in this case it fits the Application Integration pattern. We have included this pattern here, not because it is essential in this Runtime pattern, but because it is a possibility.

Guidelines for use

This pattern is only applicable if the Domino Directory is not required to store people information and, therefore, can only be used for browser-based applications. This is because Domino must have a populated directory so that it knows where to send mail to relevant people for mail routing and delivery to work.

It relies on a feature called Directory Assistance in Domino that allows for the referral of authentication requests to other Domino or LDAP directories. WebSphere Application Server fully supports using just an LDAP directory out of the box; therefore, no major changes to WebSphere Application Server deployment are required in this pattern.

Benefits and limitations

This pattern greatly simplifies deployment of WebSphere Application Server and Domino for organizations that only want to maintain one non-Domino enterprise Directory.

However, it is only of use to organizations that do not use Domino for e-mail and mail routing.

4.1.2 Using both Active Directory and Domino Directory

This pattern, shown in Figure 4-2, is most applicable for organizations that want to maintain a central Microsoft Active Directory to store user and resource information and for authentication, but still need to use Domino for e-mail.

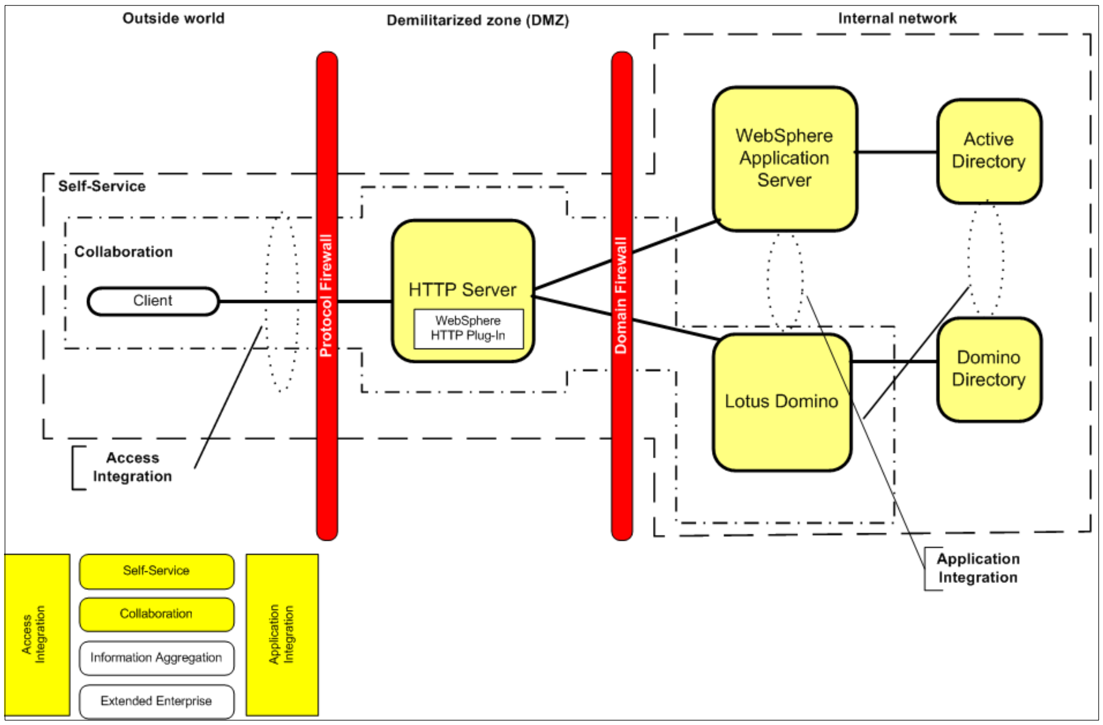


Figure 4-2 Domino and WebSphere Application Server using both Domino Directory and Active Directory.

Applicable Business patterns

The four patterns which form this topology are:

- Self-Service

  One of the user's interactions with the environment is logically with WebSphere applications. That interaction represents the Self-Service pattern.

- Collaboration

  One of the user's interactions is logically with Domino collaborative applications. That interaction represents the Collaboration pattern.

- Access Integration

  As authentication is still against a single directory (Active Directory), this is also an Access Integration pattern.

- Application Integration

  In this pattern, Domino synchronizes its user data with Active Directory using ADSync. Users authenticate against different directories according to the application they are using, but have the same username and password. As a result this is an Application Integration pattern and not an Access Integration pattern since it is not a single sign-on implementation.

Guidelines for use

This pattern is useful for organizations that do not have a policy of a single central directory, but do not wish to maintain both an Active Directory and a Domino Directory. It does not, however, address access integration, and would have to be combined with one of the other patterns to achieve this. Again, WebSphere Application Server will use either Domino or Active Directory and does not place any constraints on the implementation.

Benefits and limitations

Clearly the benefit here is reduced directory maintenance. This pattern removes the need for directory administrators to make changes to two directories when new users are added, removed, and so forth. It is also a new feature of Domino 6, and therefore does not require any third party software.

This is only a directory synchronization tool, so it does not provide any form of access integration. Solutions to that problem can be found in other patterns that can be combined with this one.

4.1.3 Using both External and Domino directories

Figure 4-3 on page 62 shows Domino and WebSphere Application Server using a central LDAP directory populated by data from other sources, including Domino using a Directory Integrator. This pattern is most applicable for organizations that want to maintain a central non-Domino directory to store user information and authenticate against but still need to use Domino for mail routing. It is the most complete solution, but it is also a little more complex.

It makes use of a Directory Integration tool that moves data from many sources into one central directory. This greatly reduces the administration overhead and effort of having a single up-to-date directory.

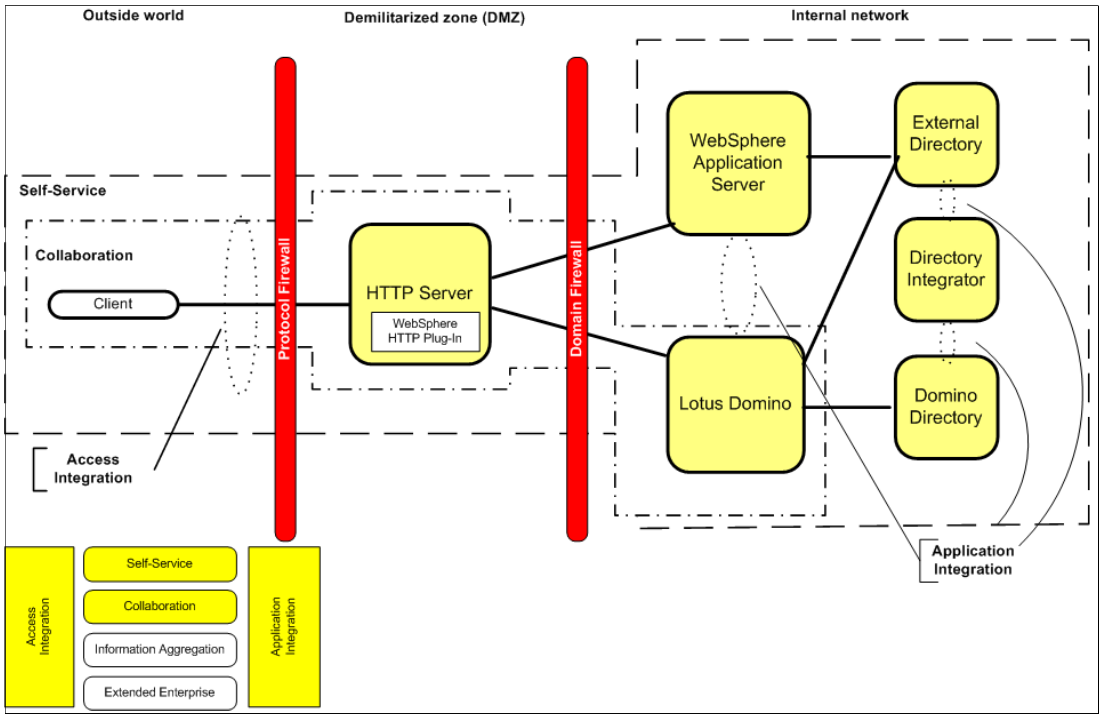


Figure 4-3 Domino and WebSphere Application Server using a central LDAP directory populated by data from both external and Domino directories

Applicable Business patterns

The four patterns which form this topology are:

- Self-Service

  One of the user's interactions with the environment is logically with WebSphere applications. That interaction represents the Self-Service pattern.

- Collaboration

  One of the user's interactions is logically with Domino collaborative applications. That interaction represents the Collaboration pattern.

- Access Integration

  By having a single directory for both application servers, this pattern fits the Access Integration pattern.

- Application Integration

  The Directory Integrator tool here brings data from many different directory and non-directory sources together into one central directory. This, therefore, is clearly an Application Integration pattern over and above the existing application integration of Domino and WebSphere Application Server.

Guidelines for use

This pattern is especially useful when several sources of people information are being maintained (directories, HR systems, telephone systems, and databases) since they can all be synchronized automatically into one directory. This can happen on an event-driven basis (a new user is added, for example) or as a batch process.

Benefits and limitations

The benefits of this pattern are realized when many different systems including a Domino and an LDAP directory need to be consolidated. It greatly reduces the administration overhead of maintaining multiple sources. It does, however, require different third party software for the Directory Integrator node.

4.1.4 Using a security server to manage authentication and connections

An alternative to using the Domino/WebSphere single sign-on feature is to use an authentication manager between the user and the servers. Tivoli Access Manager is one example of an authentication manager. It has the great advantage of being expandable to manage authentication to multiple systems at once. In more complex environments, Access Manager will enable authentication to other vendor's systems.

.

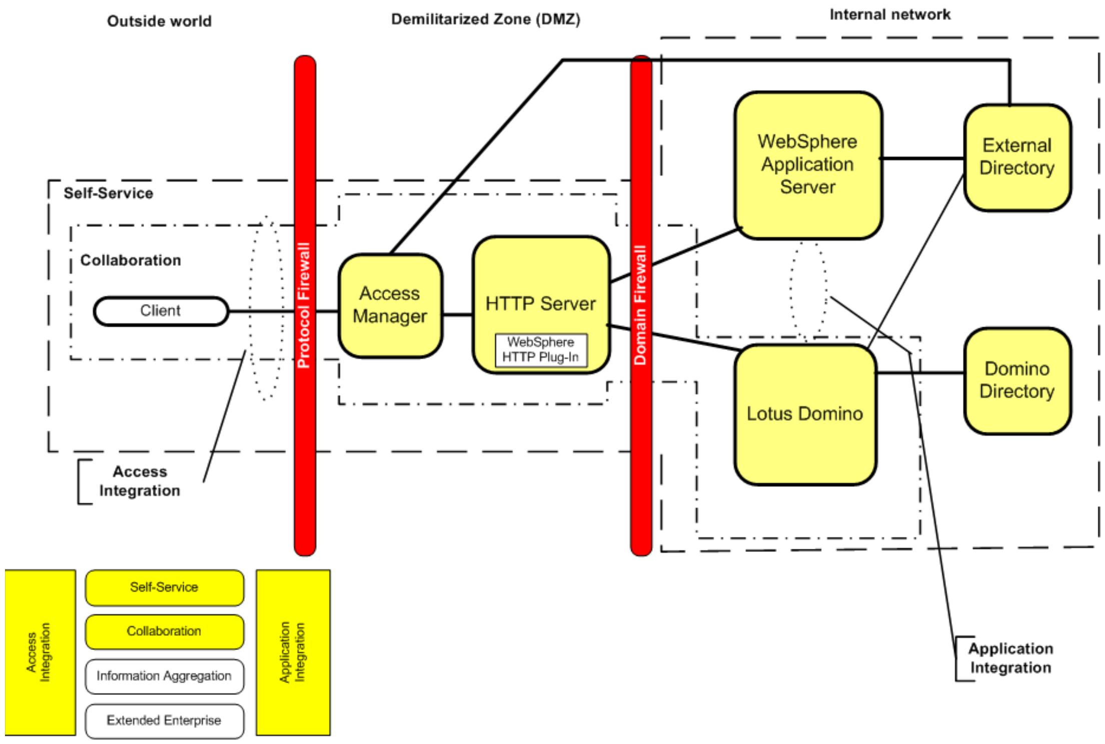


Figure 4-4 Domino and WebSphere Application Server using Access Manager as a security server

Access Manager can be configured to:

- Authenticate users via a common LDAP directory that all the servers share
- Maintain separate IDs and password credentials for each system, but hide that from the user, who is only ever prompted for the one password

The connection between Access Manager and WebSphere Application Server/Domino is via pre-defined connections (Junctions) which define the servers and their properties.

As far as WebSphere Application Server or Domino is concerned, Access Manager is just another client with an appropriate ID and password. No alterations to the applications are needed to insert Access Manager into an existing Runtime pattern. Access Manager is able to act as a reverse proxy for both the WebSphere Application Server and Domino servers simultaneously.

Applicable Business patterns

The four patterns which form this topology are:

- Self-Service

  One of the user's interactions with the environment is logically with WebSphere Application Server applications. That interaction represents the Self-Service pattern.

- Collaboration

  One of the user's interactions is logically with Domino collaborative applications. That interaction represents the Collaboration pattern.

- Access Integration

  When Access Manager provides the interface to the user, it is gathering all of the content from each of the servers that it fronts in a reverse proxy manner. Unifying the interaction with the Domino and WebSphere Application Server is an example of the Access Integration pattern.

- Application Integration

  In an environment like this, we could have Domino accessing WebSphere Application Server resources and presenting the result via Access Manager to the user; or we could have WebSphere Application Server accessing Domino resources and presenting the results via Access Manager to the client. We have included this pattern here, not because it is essential in this Runtime pattern, but because it is a possibility.

Guidelines for use

In a typical deployment of this Runtime pattern, Access Manager would reside in the DMZ, with both Domino and WebSphere Application Server behind the firewall in the internal network.

Benefits and limitations

The benefits of implementing this topology are:

- No requirements for specific versions of either WebSphere Application Server or Domino versions to enable single sign-on
- Extensible to other non-IBM/Lotus application servers
- Able to better protect the Domino and WebSphere Application Servers in the internal network
- Another layer of security that can have it's own rules and guidelines associated to achieve a slightly more granular security model

The only real limitation that this environment imposes is the cost of implementing the extra software and hardware to implement the Runtime pattern.

4.2 Domino Collaboration Runtime patterns

In looking at Collaboration patterns, probably the most basic we can picture is one with just Domino. Users could be interacting with each other via e-mail, news groups, or Web discussion forums.

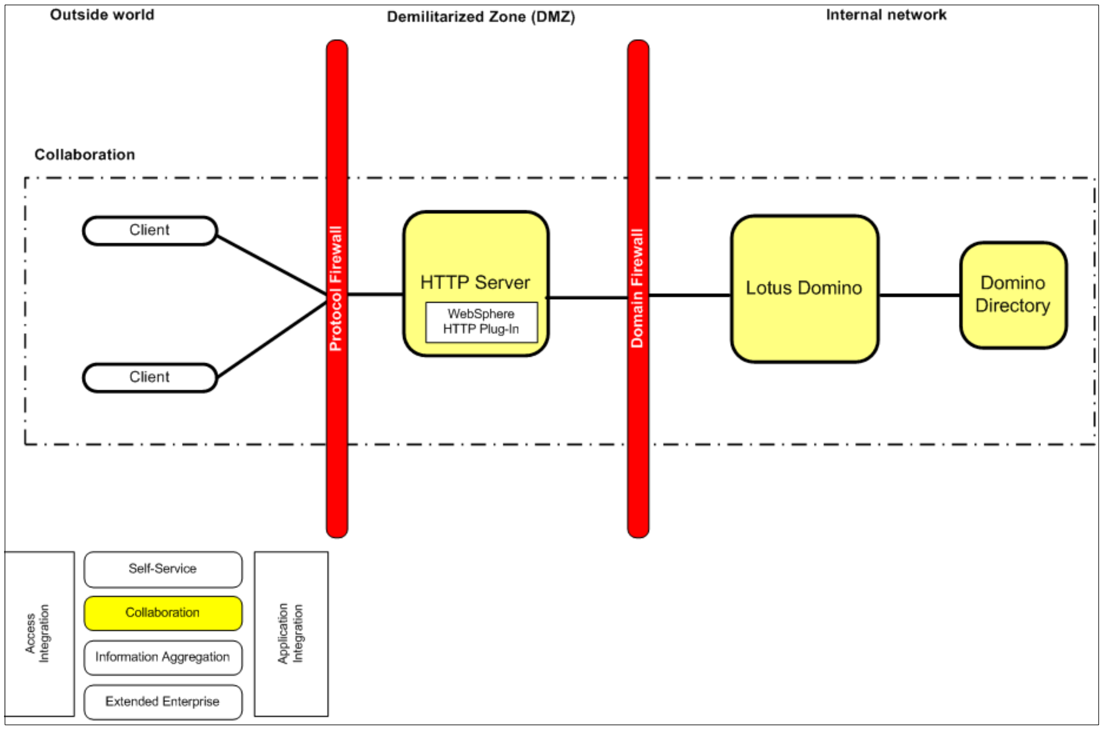


Figure 4-5 The simplest pattern using Domino for collaboration

Connections in this environment could use a number of protocols:

- NRPC - for Notes clients
- HTTP - for Web browser access to e-mail, news groups, or Web forums
- POP3/SMTP - for access to e-mails
- IMAP - for access to e-mails

Attention: Support for NNTP was removed in Domino 6.

All of these connection types represent the Collaboration pattern.

Applicable Business patterns

- Collaboration

  The only pattern at work here is the Collaboration pattern. It is Collaboration rather than Self-Service because the primary purpose of all of the interactions mentioned in the previous section is to communicate with other users.

Guidelines for use

It should be noted that this pattern makes use of the WebSphere HTTP plug-in since it allows the Domino server be located behind the firewall. This is recommended in even the simplest Internet-facing installations to prevent data from residing in the DMZ.

Benefits and limitations

Being a straightforward installation, configuration and management is relatively easy. Maintenance is also simple.

4.3 Sametime collaboration topology

While the previous topologies dealt with *asynchronous* collaboration interaction, incorporating Sametime introduces the idea of *synchronous* collaboration interaction. Since Sametime will only work when installed on top of Domino, the previous patterns of directory architecture still apply to patterns involving Sametime. All the patterns in this section assume that Sametime uses an external directory.

4.3.1 Sametime only

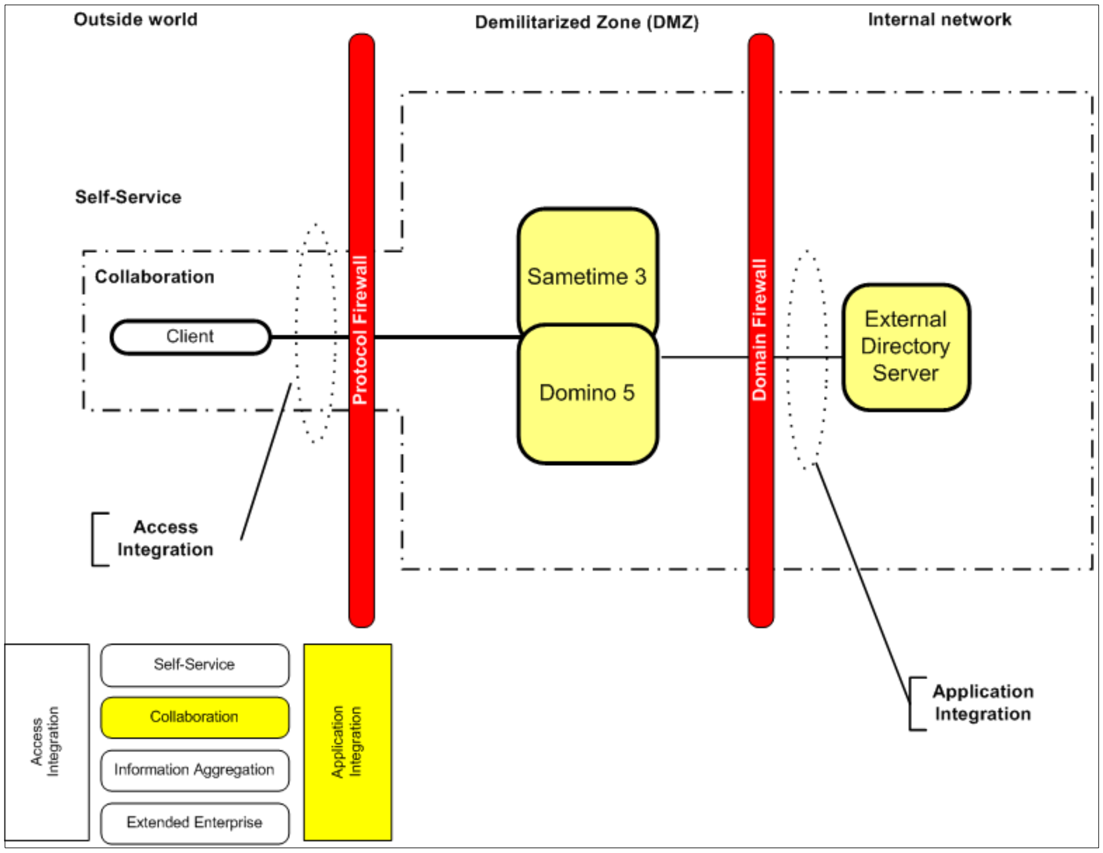


Figure 4-6 A simple Sametime Collaboration pattern

Synchronous collaboration is provided by Sametime 3, which will only run on top of Domino 5.0.10. The option to install a stand-alone version of Sametime was removed with version 3. While it is possible to use the underlying Domino server for collaboration and messaging, this is not a recommended deployment since this can adversely affect the performance of Sametime.

Connections to the Sametime server from the various client types (Sametime Connect, Sametime Meeting client, Broadcast client) are via the appropriate protocols on the configured TCP/IP ports.

Applicable Business patterns

- Collaboration

  The main pattern at work here is the Collaboration pattern. It is Collaboration rather than Self-Service because the primary purpose of all of the interactions (one-to-one, one-to-many, or many-to-many) is to communicate with other users.

- Application Integration

  Since we have chosen to have an external directory, the Application Integration pattern is also apparent.

Guidelines for use

For external users, this Runtime pattern represents a fairly standard Sametime installation with Sametime residing in the DMZ typically. Internal users normally have their own internal Sametime server that they use. This internal server communicates with the DMZ Sametime server. This pattern can be further enhanced using the SIP Gateway extension to Sametime, which allows enterprises with separate Sametime communities to collaborate with each other.

Benefits and limitations

Sametime 3 introduced support for the LTPA token via the Browser Client, which greatly simplifies the end-user experience by avoiding the need to log in twice.

It is difficult with this basic Runtime pattern to move the Sametime server to the internal network and still permit external users to interact with the server. This is because Domino 5 does not support the WebSphere HTTP plug-in. The server needs to reside within the DMZ and is therefore more at risk.

It is also important to note that with Sametime 3, there is no longer the option of a standalone version. Domino must be installed and configured before Sametime can be installed.

4.3.2 Sametime and Domino 6

In this pattern, asynchronous collaboration is provided by Domino and synchronous collaboration is provided by Sametime. Sametime 3 will only run on top of Domino 5.0.10, so any pattern that requires Sametime 3 and Domino 6 will require two versions of Domino.

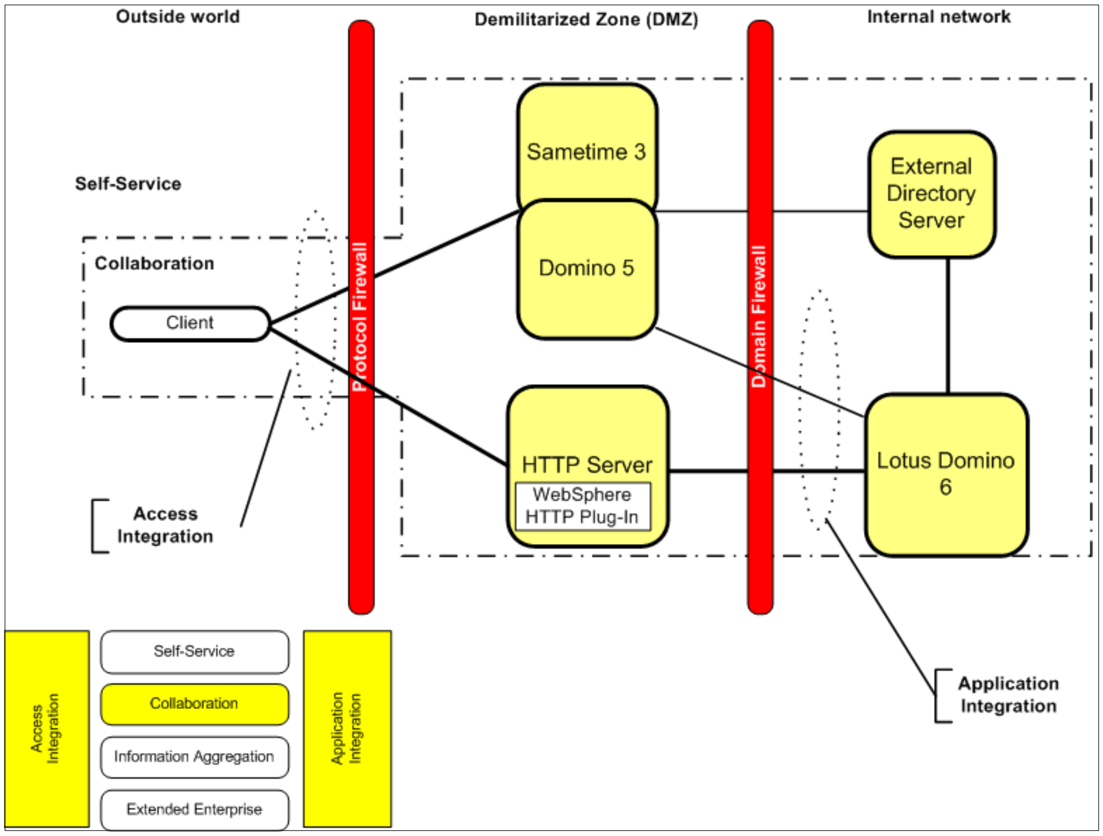


Figure 4-7 A collaborative pattern using Domino and Sametime

Connections to the Sametime server from the various client types (Sametime Connect, Sametime Meeting client, Broadcast client) are via the appropriate protocols on the configured TCP/IP ports.

Connections to the Domino server could be for HTTP sessions, but could also be for Notes clients, POP3 or IMAP clients.

Applicable Business patterns

The three patterns at work in this topology are:

- Collaboration

  This type of topology is usually used in environments where you want to incorporate instant messaging with existing Domino environments and security. Since the primary mission for the server is collaboration interaction, the Collaboration pattern is involved here. The collaborative functions of the Lotus Domino 6 server can be used for user-to-user interaction.

  The Domino 6 server can also be used for Self-Service patterns. We have not added that to the diagram for the sake of clarity. Technically, the Sametime server could be used for Self-Service patterns as well. We have not added that to the diagram mainly because it is not recommended to use a Sametime server in that manner.

- Access Integration

  In a combined Domino/Sametime application, the use of single sign-on using the LTPA token prevents the user from being prompted multiple times when switching between an application that is served by Domino and services provided by Sametime. The single sign-on works in the same manner as single sign-on between Domino and WebSphere, being dependent on a token that the user's browser holds.

- Application Integration

  Integration between Sametime services and Domino applications on this Domino server or other Domino servers is an example of application integration. Examples of this type of integration include providing "who is on-line" or "who is here" functionality to a Domino application that is served by Domino to Web-based users. The level of application integration might also extend to the application initiating a Sametime conversation between a help desk operator and the user.

Guidelines for use

For external users, Sametime implemented in this Runtime pattern would normally reside within the DMZ. In order to have internal users interact with the external users, an internal Sametime sever would normally be installed.

This Runtime pattern will enable you to create applications with user awareness built in. For more details and real-life scenarios illustrating this pattern, see the IBM Redbooks: *B2B Collaborative Commerce with Sametime, QuickPlace and WebSphere Commerce Suite*, SG24-6218; *Lotus Sametime Application Development Guide*, SG24-5651; *Working with the Sametime Client Toolkits*, SG24-6666; and *Working with the Sametime Community Server Toolkit*, SG24-6667.

Benefits and limitations

Sametime servers implemented in this manner can be readily integrated into more complex Domino and WebSphere environments, sharing the same single sign-on token between all of them. In that way, users will only ever be prompted once for their ID and password.

Like the other Sametime topologies, this Sametime server needs to reside within the DMZ for external users to attach to it. Internal users will need their own internal Sametime server, as well, if they are to interact with the external users.

At the time of writing, Sametime 3 was only supported on Domino 5.0.10, meaning that different versions of Domino would have to be installed to get the benefits of both products.

4.4 Single sign-on solutions

Here we consider three situations where single sign-on is needed but we can use two technologies to provide that functionality. The first is when just Domino, WebSphere, and Sametime are required technologies, and we can rely on the LTPA support that is provided in all three applications.

There will be times where single sign-on is required for solutions that integrate with applications that do not support LTPA; in these cases we must rely on an authentication manager such as Tivoli Access Manager.

When we combine WebSphere with Domino and Sametime into a topology we find that we are extending the Runtime patterns discussed in 3.4, "Utilizing IBM single sign-on" on page 55 and 4.1.4, "Using a security server to manage authentication and connections" on page 63.

4.4.1 Single sign-on with Sametime, Domino, and WebSphere

Like the two mentioned in the previous paragraph, this Runtime pattern (illustrated in Figure 4-8) is dependent on implementing the single sign-on between all three servers.

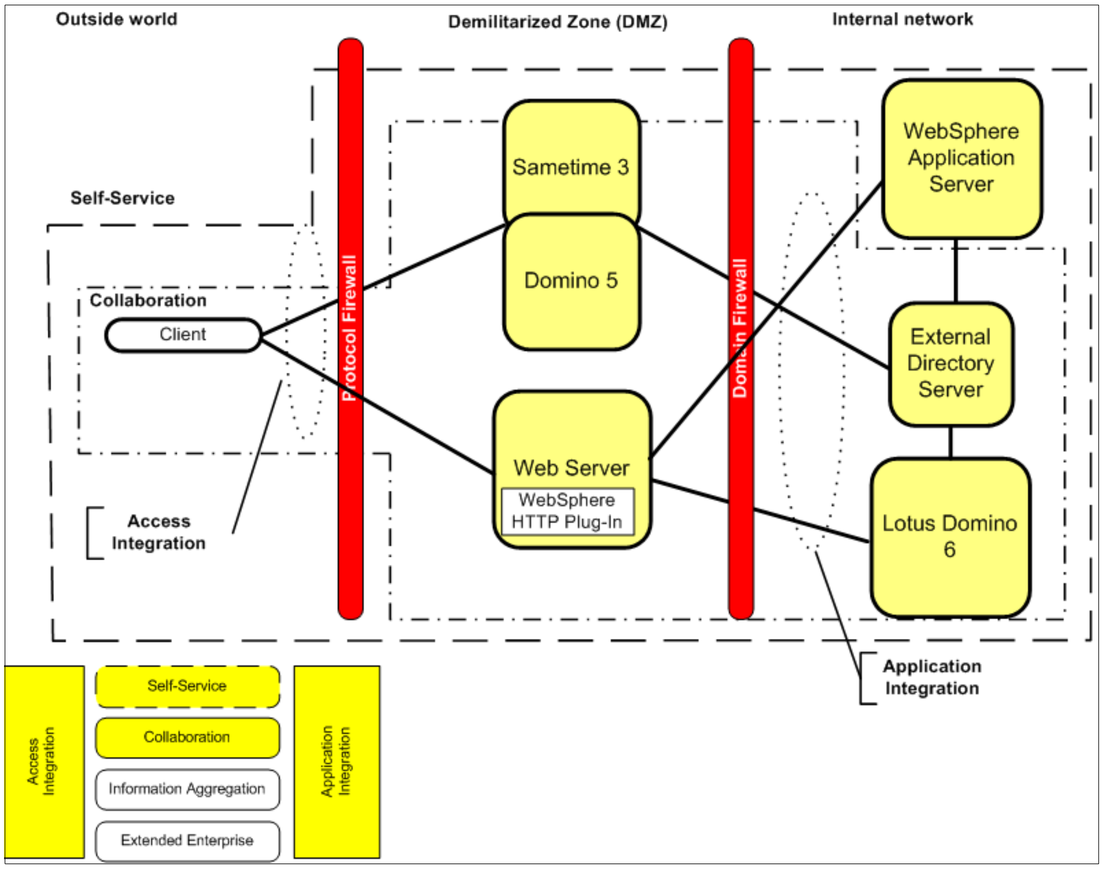


Figure 4-8 A collaborative pattern using Domino and Sametime with single sign-on using LTPA tokens

We are starting to get into quite complex Runtime patterns, which can be used to implement powerful Web environments with many uses. While WebSphere can be used for collaboration interaction, in this topology, the better tools for the job are the Domino and Sametime servers.

This topology would typically require the Sametime server and the HTTP server (for Domino 6 and WebSphere 5) to be in the DMZ.

Applicable Business patterns

The patterns at work in this topology are:

- Collaboration

  With Sametime and Domino providing synchronous and asynchronous collaboration interaction, these two servers along with the users form the core of the Collaboration pattern in this topology.

- Self-Service

  By adding WebSphere into this topology, we have implied more than just collaboration interaction and we have effectively added some level of Self-Service integration. While it is possible to do that without WebSphere and with just Domino, we assume that the application calls for a more transactional design than Domino is ideally suited to. We have excluded Sametime from this pattern because it is really only for collaboration interaction.

While it is possible to implement Sametime “bots” that automatically respond to a user’s questions, most Sametime implementations are aimed at collaboration interaction and the automated responses to user questions can easily be achieved with either Domino or WebSphere. If Sametime bots were more common, then Sametime would probably be included in the Self-Service pattern.

- Access Integration

  Provided the implementation prerequisites have been met, access integration occurs on the user’s machine with all three servers. The client machine is a key part of the access integration because the authentication token that is valid for all three servers is stored on their machine.

- Application Integration

  The application integration in this Runtime pattern could be between Domino, Sametime, and WebSphere. Examples of this are presented in the Sametime-related Redbooks cited previously.

Guidelines for use

To implement this topology for external users, Sametime and either Domino or WebSphere would normally reside within the DMZ. The third server (Domino or WebSphere) could also reside in the DMZ, or it could be in the internal network behind the firewall. This is because an external Sametime server cannot be used to serve the internal users, so whichever server is going to interact with the users should reside in the DMZ.

In order for the Sametime server to share it’s single sign-on token with both the Domino and WebSphere servers, Sametime must be installed on top of a Domino 5.05 (or later) server. Users would then be able to authenticate with Domino, for instance, and then begin using Sametime with the Sametime server, knowing who the users are and that they are entitled to use Sametime.

Benefits and limitations

One of the benefits of this solution to the multiple server authentication problem is its simplicity for both administrators and users. Once users have authenticated and hold a token, they can easily access resources on the other server without the need to key in their ID and password again.

The Runtime pattern has some limitations, specifically:

- The Sametime server and related Domino server need to reside within the DMZ for external users to attach to them. Internal users will need their own internal servers, as well, if they are to interact with the external users. Having servers residing in the DMZ puts them at more risk that internal servers.
- In order to support the single sign-on, users must support cookies because the authentication token is stored on the client inside a cookie.
- It is not extensible to non-IBM/Lotus application servers.

4.4.2 Sametime and Domino with Tivoli Access Manager

By inserting Access Manager between the users and the servers, as illustrated in Figure 4-9, we can take advantage of Access Manager’s abilities to provide single sign-on across multiple disparate systems.

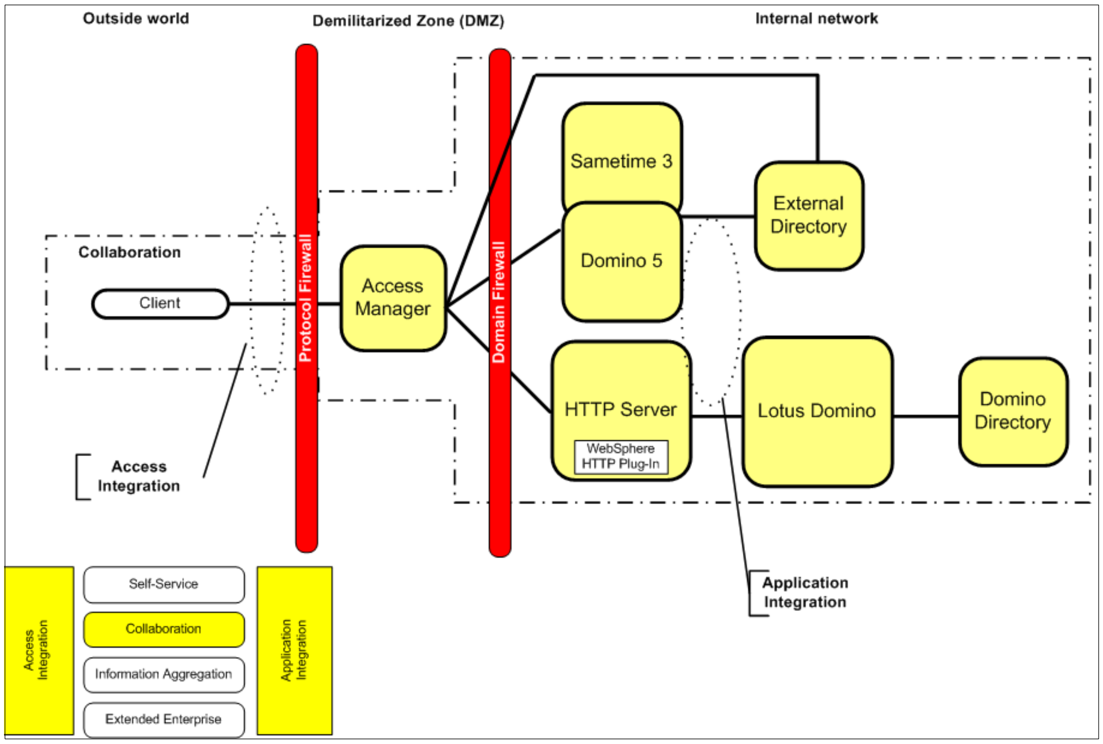


Figure 4-9 A collaborative pattern with Domino and Sametime using Tivoli Access Manger

Here, Access Manager is acting as a reverse proxy for all client access.

Exactly the same application integration that occurred in the previous topology will apply with this Runtime pattern as well. The only difference with this topology is how the access integration is implemented.

Applicable Business patterns

The patterns involved in this Runtime pattern do not depend on how Sametime is installed, as they did in the previous Runtime pattern. The patterns are:

- Access Integration

  Access Integration in this Runtime pattern makes use of Access Manager's global sign-on feature to integrate the access to both the Sametime and Domino servers without requiring the user to enter their ID and password again. Therefore, we have access integration occurring between Access Manager, WebSphere Applications Server (the HTTP Server), Sametime, and Domino.

- Application Integration

  This topology is an example of application integration when the application on the Domino server exhibits "who is here" or "who is online" features.

- Collaboration

  The Collaboration pattern is involved here because the main aim of the environment is to enable users to interact with other users.

Guidelines for use

In a typical deployment of this topology, Access Manager would reside in the DMZ, with both Domino and Sametime behind the firewall in the internal network.

Since the Sametime server is now on the internal network and therefore more secure, internal users can access it directly, which eliminates the requirement to have additional Sametime servers (provided they have the spare capacity, of course).

Benefits and limitations

The benefits of implementing this topology are:

- It is extensible to other non-IBM/Lotus application servers.
- It is able to better protect the Domino and Sametime servers in the internal network.
- It provides another layer of security that can have it’s own rules and guidelines associated, to give a slightly more granular security model.

The only real limitation that this environment imposes is the cost of the extra software and hardware to implement the topology.

4.4.3 Combined Sametime, Domino, and WebSphere with Access Manager

This Runtime pattern (Figure 4-10) is similar to the previous one except we are no longer making use of the Domino/WebSphere single sign-on token.

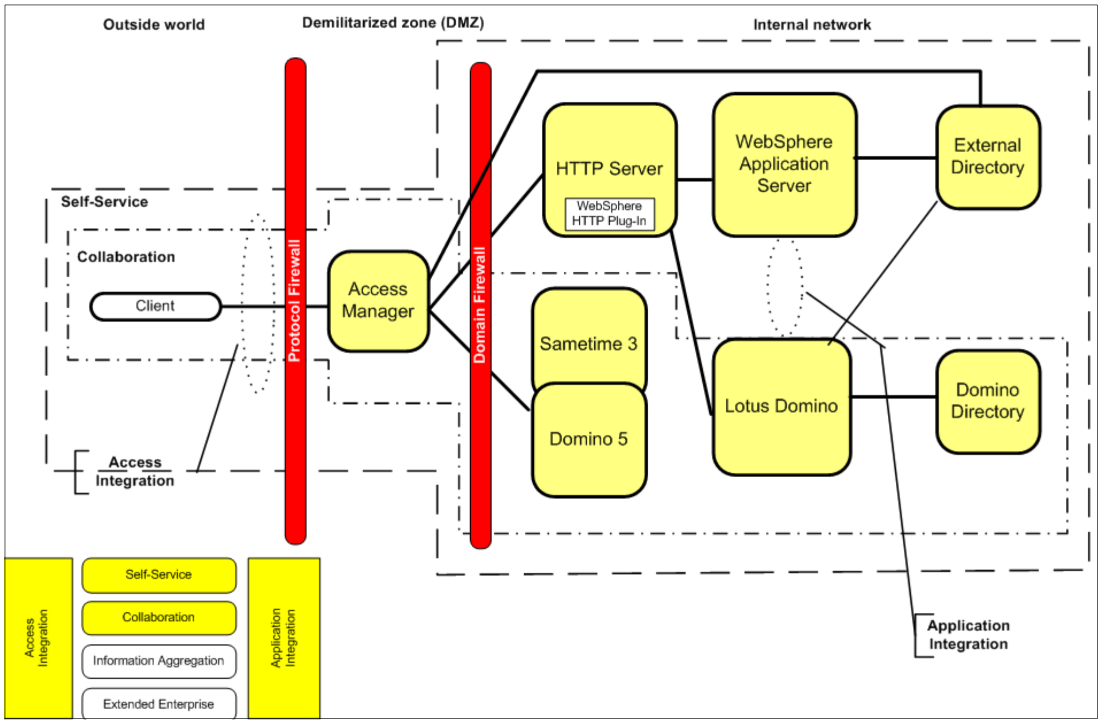


Figure 4-10 A collaborative pattern using Domino, Sametime, and WebSphere using Tivoli Access Manger

Not being dependent on the Domino/WebSphere single sign-on token has a number of implications:

1. Sametime no longer needs to be installed as on overlay on an existing Domino installation.
2. All of the servers (Sametime, Domino and WebSphere) can be moved back into the internal network (or second level DMZ), with only the Access Manager in the DMZ.
3. The Domino server could potentially be any version from R4.5 onward and still be able to maintain the global sign-on integration.

4. WebSphere could be at any version and still be able to maintain the global sign-on integration.
5. Users no longer need to support cookies in order to authenticate across multiple systems.

Applicable Business patterns

The patterns associated with this topology are:

- Collaboration

  With Sametime and Domino providing synchronous and asynchronous collaboration interaction, these two servers are enabling all of the collaboration interaction.

- Self-Service

  With WebSphere in this topology, we have implied more than just collaboration interaction. A common reason for inserting WebSphere into the topology is to add some level of Self-Service integration. While it is possible to do that without WebSphere, and with just Domino, we assume that the application calls for a more transactional design than Domino is ideally suited to.

- Access Integration

  Access Integration occurs in this Runtime pattern between the servers and Access Manager. The user is no longer involved in the access integration since Access Manager is performing all of the integration with regard to authentication. The user no longer needs to accept the authentication token.

- Application Integration

  The application integration in this topology could be between Domino, Sametime, and WebSphere. Examples of this are included in the Sametime-related Redbooks cited previously.

Guidelines for use

In a typical deployment of this topology, Access Manager would reside in the DMZ, with Domino, Sametime, and WebSphere behind the firewall in the internal network.

Since the Sametime server is now on the internal network, internal users can access it directly, which eliminates the requirement to have additional Sametime servers (provided they have the spare capacity, of course).

Benefits and limitations

The benefits of implementing this topology are:

- It is extensible to other non-IBM/Lotus application servers.
- The Domino and WebSphere servers are better protected in the internal network.
- It provides another layer of security that can have it’s own rules and guidelines associated, to give a slightly more granular security model.

The only real limitation that this environment imposes is the cost of the extra software and hardware to implement the topology.

5

Choosing Domino-WebSphere Hybrid Runtime patterns

As we move further into the application of the Patterns for e-business, we must arrive at product selection. In order to get there, we need to develop criteria for evaluation. These evaluation criterion will serve as the basis for making decisions when producing a solution based on Domino and WebSphere.

There are few instances where there is a *de facto* solution because all environments are different. Guidelines outlined in this chapter must be tailored to the specific needs of each environment. Runtime patterns with Domino and WebSphere do not seek to reproduce the process of technology selection. The book *Patterns for e-business: A Strategy for Reuse*, provides a direction in terms of mapping products to the solution.

The intent is to provide some evaluation criteria that would be applicable in most technology deployments where Domino and WebSphere are being compared. Refer to Chapter 3, “Lotus Domino and WebSphere Application Server patterns” on page 31 for an in-depth discussion about the specific benefits and limitations of each technology.

5.1 Comparison guidelines

There are a number of factors that will affect the decision to use Domino, WebSphere, or both. There are also decisions to be made in the lifecycle of a system, as to whether or not a technology change is needed. It is also true that there is a significant level of overlap between Lotus Domino and IBM WebSphere Application Server.

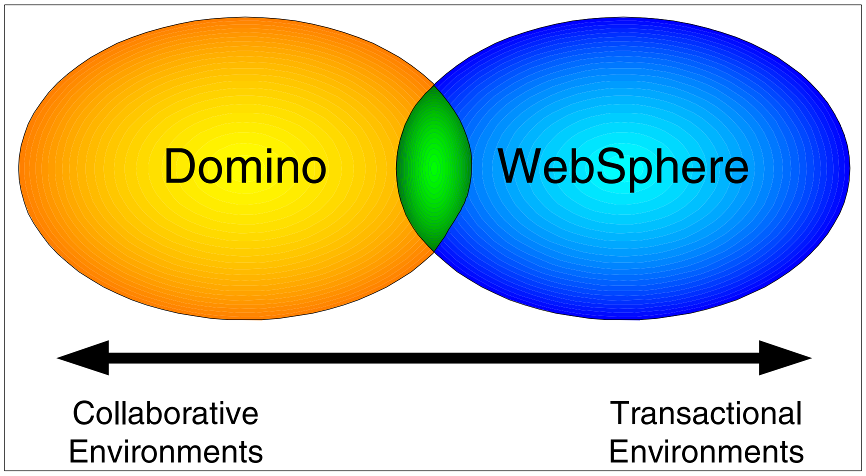


Figure 5-1 Domino and WebSphere crossover

Historically, confusion about product selection has centered around where to use which technology. However, the answer really lies in the analysis to be completed *before* arriving at product selection. The Patterns for e-business are technology agnostic. However, when implementing Application patterns with the Hybrid Runtime patterns found in Chapter 3, those Runtime patterns will exhibit service level characteristics, which include:

- Availability
- Performance
- Scalability
- Security

We have tried to present some guidelines for decision making within your organization. As each organization is different, by no means should the guidelines here be considered final decision criteria. They are meant only to provoke thought for your decision process.

5.1.1 Navigating the Hybrid Runtime patterns

There are two methods you can use to navigate the Hybrid Runtime patterns. The first is simply to move through them in the linear order in which they are presented. This method provides an in-depth review of all the WebSphere/Domino integration designs.

If you already have an understanding of the required functionality, complexity, and extensibility of the solution you're designing, use of the following table is recommended. It divides the WebSphere/Domino Integration designs into five categories based on focus of functionality. It then further divides the designs according to the complexity and extensibility of each.

By *Collaboration-centric*, we mean Custom designs which provide mostly functionality related to collaboration through the Domino server. The Custom designs include some level of functionality where WebSphere resources are accessed; if not, we categorize the Custom design as *Collaboration only.*

With *Transaction-centric*, we mean Custom designs which provide mostly functionality through the WebSphere Application Server. The Custom designs include some level of functionality where Domino resources are accessed; if not, we categorize the Custom design as *Transaction only.*

Custom designs where services are provided by the Domino server and WebSphere Application Server somewhat equally, or could as well provide either, are categorized as *Dual focus*.

Note: We have intentionally left the *Transaction only* category off the table because Custom designs in that category are beyond the scope of this redbook. Instead, refer to the Patterns for e-business Web site at:

`http://www.ibm.com/developerworks/patterns/`

Various IBM Redbooks (for example, *Patterns: Self-Service Application Solutions Using WebSphere Application Server V5,* SG24-6591) provide additional information about Self-Service Business patterns.

The numbers in the table refer to the figure numbers where the recommended topologies are illustrated. (For example "3-5" refers to the fifth figure in chapter 3.)

Table 5-1 Custom designs for Domino and WebSphere integration categorized

	Entry level	**More advanced**	**More scalable and available**
Collaboration-centric	3-3, 3-4	3-5	*
Transaction-centric	3-6	3-5, 3-7	*
Dual focus	3-5, 4-1	4-2, 4-3, 4-4, 4-8, 4-10	*
Collaboration only	4-5, 4-6	4-7, 4-9	9-3, 9-4

* See the IBM Redbook, *Patterns for the Edge of Network*, SG24-6822, for details.

5.1.2 Organizational structure

The Patterns for e-business help to identify the interaction between users and businesses. The patterns provide valuable information to develop your e-business application.

However, before you begin using the patterns, the organization should have a firm understanding of the following:

- Business problem to be solved
- Current application and data configuration
- Installed Runtime products
- Available skills
- Quality of service required

No technology should be implemented without consideration of the impact on the organization and its people, processes, and technology investments made to date. Frequently, these topic areas are discussed as a comparison of business and IT drivers. The decision between Domino and WebSphere may further be illuminated in the light of some of the following areas, regardless of categorization.

Agility

In a business context, *agility* refers to the ability to rapidly adapt to a changing environment. How agile an organization requires the solution to be is one of many criteria you should consider. Some questions to help determine this are:

- Does the nature of the business require frequent changes in systems?
- Have the core business processes been clearly identified and documented?
- Will the system be required to interact with other external customers?

Domino historically has a shorter development cycle than WebSphere and its Java (J2EE) development environment. Agility can be built into an individual application if certain variables are known in advance. However, if the business requires significant change, one way of measuring this might be to look at development cycles. Typically, an application developed on a Domino platform has a shorter development cycle than does an application developed for WebSphere.

Note: This situation may not always be the case for all applications, and it is also highly dependent on other requirements placed on the system.

Time to market

Across industries there are products that have varying life cycles. The window of opportunity for some products is short while other products can continue for many years. Also, some industries spring up very quickly while others take years to mature. The IT systems that support these contrasting environments will also have advantages or disadvantages. Some questions to help determine the *time to market* might include:

- Does the business product have a short life cycle?
- Does the business product being supported require speed to market?

Domino historically fares well in organizations where there is rapid turnover. This is due to its short development cycle and lower initial cost of entry. Domino is advantageous for solutions requiring rapid development and deployment.

However, longer life cycles tend to favor WebSphere, where the solution is more likely to provide a higher return on investment.

Strategic intent

It is important to understand what business strategies a company is employing. IT systems need to be able to scale to accommodate growth and provide flexibility to execute the company’s strategies. Some questions to help determine the *strategic intent* might include:

- What is the ultimate goal of the company?
- What are the growth projections?
- What is the execution timeframe?

Understanding the company’s value proposition will provide insight as to how the IT solutions for the company should be architected. If a company is executing strategies to become a large on-line vendor with significant marketing efforts to drive traffic to the site, this large-scale scenario would tend to favor WebSphere over Domino since WebSphere has greater ability to scale.

Employee work modes

In executing its strategies, a company will have employees executing tasks. This question relates to the division of that labor. Some questions to help determine the impact of *employee work modes* might include:

- Are the employees mobile?
- Do they normally work while in transit?
- At what speed are employees required to complete business transactions?
- Do employees have the ability to connect to the home office?

If employees are more prone to work in a stand-alone mode, this clearly favors Domino over WebSphere.

Since its inception, the Lotus Notes/Domino product line has been designed around granting mobility to the end user. The replication process enables this feature within Lotus Notes/Domino. While custom application could provide some of these services within WebSphere, there are no equivalent technologies within WebSphere out of the box, so the advantage is with Lotus Domino.

However, more transactional types of interactions would require a solution to add pervasive device capabilities. Both Domino and WebSphere have this ability.

Some situations may require both transactional and stand-alone features. In this scenario, there could be business cases made for both Domino and WebSphere to be deployed together.

IT environment

It is worth a look at the *IT environment* when making a decision between Domino and WebSphere. The structure of the current IT organization and its skill levels will be vital to creating and maintaining the solution. Among the questions to consider are:

- How good is the relationship between IT and the business it supports?
- Is the IT organization clearly structured in terms of process and development?
- Has the IT staff delivered in large projects?
- Does the IT have clearly defined standards?
- Are the existing standards and procedures adequately enforced?
- Can the organization attract and retain qualified personnel?

While there are many questions to probe this topic, the takeaway should be a view of the organization in terms of the degree of organization in the IT environment. A Domino solution would tend to fit better into a more loosely organized environment. A WebSphere solution places a higher standard upon the IT organization in that the IT organization must have clear processes, procedures, and qualified personnel successfully executing them.

Workflow requirements

This issue, like the employee work modes, relates to the division of labor. Some questions to help determine the impact of *workflow requirements* might include:

- Do business processes routinely span multiple departments?
- Are business processes clearly defined?
- What are the technology dependencies in affected departments?

Lotus Domino has a tremendous record of success in creating workflow applications. The development tools are historically designed to facilitate this end. This does not mean that WebSphere cannot be adapted to perform the same workflow tasks. WebSphere, on the whole, will take more development effort and incur higher on-going costs than Domino, with all other things being equal.

Data requirements

On transactional systems, you typically need to be absolutely sure that a change has been applied to a database when updated. This referential integrity is normally found within relational database management systems (RDBMS), such as DB2. Domino database is not a relational database and is not well suited to storing large volumes of data that has links to other data, such as in the tables of a relational database.

Domino addresses this issue by providing connection tools such as the following:

- DECS for real time connections
- Lotus Enterprise Integrator for real time or scheduled tasks
- LSX libraries for real time or scheduled tasks
- ODBC Connectors for real-time or scheduled tasks
- JDBC Connections for real-time or scheduled tasks based on Java agents or code embedded in Domino applications

Domino has a capability, via Lotus Enterprise Integrator (LEI), to store all data in a DB2 database instead of in a Domino database. The benefit of this is that Domino would have all the security controls and database design, but all data is actually stored in DB2. This access method is called Advanced RealTime. Prior to Domino 6, storing the data in a DB2 database required stub documents to be stored in Domino. Now, all data is stored in DB2, but it can be viewed and used in Domino. The documents are referred to as *virtual documents*. There are also *virtual attachments*, which allow you to store the attachments in an external system, while they appear in Domino as regular document attachments, and *virtual agents*, which allow you to call programs in an external data source. Advanced RealTime connectivity allows significantly better scalability and performance since there is no need to store data, not even “key documents,” in a Domino database.

It should also be noted that when Domino stores data it normally stores that data locally inside its own database (NSF file) format. Data read/write operations are subject to the I/O performance and costs associated therein.

WebSphere can access relational data stores directly using JDBC or EJB interfaces. If we consider the various options for interacting with a relational database, due to the transactional nature of WebSphere, we can achieve high performance by employing techniques such as connection pooling, caching, and minimizing JNDI lookups.

WebSphere also allows for the abstraction of databases and their connections from application code by accessing a business object layer rather than the database directly. This layer, in turn, accesses the database itself. Therefore, many different applications can access the same business objects and not necessarily be part of the same application. This decouples the database design from the application itself, creating easier maintenance going forward.

WebSphere insures referential integrity by supporting two-phase commits. This increases data integrity. Therefore for example, it can be guaranteed that a purchase order received and the payment details are correctly stored in the appropriate databases.

For solutions that only occasionally need access to relational data, Domino methods can be employed to lookups or updates. However, more transactional types of interactions are better suited for WebSphere.

Operational availability

Availability is also a highly important variable. There are different methods employed to provide scalability. However, the right solution depends upon other factors, such as transactional volume and cost. A thorough discussion of the various mechanisms for achieving scalability and redundancy is found in Chapter 9, “Scalability and redundancy” on page 161.

Transactional volumes

With respect to transactional volumes, WebSphere, with its ability to be spread across multiple nodes, is a distinct advantage over Domino, which must be deployed in total on a physical node. WebSphere also has superior EJB containers that allow for better integration with the data layer.

When comparing transactional volume it is common knowledge that WebSphere can be scaled to meet higher system requirements than can Domino, given equivalent platforms.

Target audience

Along similar lines to transactional volume, the audience being targeted by the system is very important. Pertinent questions are:

- Are the primary targeted users Internet users?
- Is the system to be open to all users or the Internet?
- Is the system designed for a defined subset of users such employees, customers, or vendors?
- Will there be any significant growth in the foreseeable future?
- Are there marketing campaigns designed to bring visitors to the site?

Domino has found good success supporting intranet solutions and extranet solutions. Domino has also succeeded supporting Internet solutions when other factors—such as transactional volume, application complexity, data integration needs, and operational requirements—are not too highly taxing on the hardware deployed. If the size and scope of the solution is a relative known factor, such as in an intranet solution, this simplifies things: Domino hardware can be sized accordingly. In general, larger sites, data volumes, and greater application complexities tend to favor WebSphere.

Costs and revenues

IT exists to save costs and drive revenues and ultimately these are the driving factors in any organization. While this can be measured many ways, and data can be gathered by various means, the basic themes are true:

- The IT solution should minimize marginal costs.
- If two IT solutions are being compared, the optimal solution would be the one with lower marginal costs over the time period being measured.
- Marginal revenue should influence the solution sizing.

A comparison of the costs involved in creating a solution based on WebSphere shows that it has greater development time, more skillsets required, and greater infrastructure needs, which translates into greater costs. It can be assumed that the initial cost of entry (initial marginal cost) for WebSphere is relatively high, whereas Domino is lower. However, the cost per transaction is generally lower for WebSphere. Since there is greater demand placed on the system, and the solution produces a greater quantity of transactions, the marginal cost for maintaining a Domino system will be higher, in general.

Unfortunately, determining the definitive point where the marginal cost of one solution is greater than the marginal cost of the other solution is not an easy task. This point is hidden by the fact that each solution is unique.

Additionally, solutions can be leveraged to support multiple applications and processes. Supporting a number of processes with a single application or service can lower the total cost of applications.

5.2 Product alignment to system tasks and attributes

Another aspect of the decision between Domino and WebSphere is how well the product aligns with the task it is being asked to perform.

Table 5-2 provides a general ranking of how well Domino and WebSphere align to a list of commonly encountered tasks or system attributes.

Table 5-2 Domino and WebSphere suitability to specific tasks

Task/Attribute	**Domino**	**WebSphere**	**Comments**
Object types	Moderate	Excellent	Domino uses a proprietary object model featuring items such as forms, views, documents, agents, pages, and databases, and has adopted Java. WebSphere uses the open development language Java via Servlets, Java Beans, Gasps, and EJBs, and thus is limited more by language and programming feasibility. Domino 6 and future versions will open the proprietary Domino Object Model to other tools. The technologies used are XML for data transfer and Domino tag libraries for exposing the Domino Object Model in a JSP environment.
Scalability	Good	Excellent	Domino systems can become large, while WebSphere can scale to massive sizes.
Skills required	Good	Fair	Domino developers require moderate to high skillsets, but a power user can create productive applications. WebSphere developers require high skill levels and power users cannot readily participate in creating applications.
Development models	Good	Good	Domino can use Formula language, Lotus Script, BASIC, COM/COM+, Java, J2EE and more. WebSphere is limited to the Java (J2EE) model.
Clients supported	Excellent	Very Good	Domino supports Notes, browsers, mail clients, and LDAP clients. WebSphere supports browsers.
Protocols supported	Excellent	Good	Domino supports NRPC, HTTP, IIOP, SMTP, IMAP, POP3, LDAP, and more. WebSphere primarily supports HTTP and IIOP.
Application tools	Moderate	Good	Domino can use Domino Designer, WebSphere Studio, and third-party tools. WebSphere developers have a variety of tools, such as WebSphere Studio, VA Java, and more.
e-Mail	Very Good	NA	Domino is a market leader for e-mail. This functionality is not native to WebSphere.

Task/Attribute	Domino	WebSphere	Comments
Discussion forums	Good	Poor	Domino has database templates dedicated to this purpose. WebSphere would require a relatively high amount of programming to achieve this functionality.
Distributed content creation and approvals	Good	Good	WebSphere requires additional software and configuration. Domino requires workflow application development or third-party applications.
People-oriented workflow	Good	Moderate	WebSphere would require significant programming effort. Some templates that ship with Domino include workflow capabilities. Workflow functionality can be added to any Domino application without significant programming effort.
Rapid Application Development	Good	Moderate	WebSphere would require significant programming effort.
Integration with other systems	Good	Good	Both Domino and WebSphere do well. Domino integration is typically simpler to install and configure, whereas WebSphere integration tends to be more robust.
Highly scalable systems integration	Moderate	Good	Both Domino and WebSphere are scalable vertically and horizontally. Domino tends to be less scalable due to the overhead of server tasks which make it a heavier server. The Domino server is therefore more closely tied to the addition of memory and processing power. WebSphere is designed to perform jobs it is programmed to do, making it lighter. WebSphere also, by design, makes it simpler to distribute processing tasks across multiple pieces of hardware.
Transactional processing	Poor	Good	The Domino data store excels in document-based computing, but not in transactional processing.
Relational data	Fair	Good (via DB2)	Domino touches relational data only via use of connector tools. However, it is possible to use a relational database as the data store via LEI. WebSphere connects to DB2 databases via EJBs or JDBC.
Java development	Good	Excellent	Java development in Domino tends to be centered around the Domino programming model. Java development is currently more robust in WebSphere. New in Domino 6 is the ability to instantiate Java objects, call Java objects methods, and read/write Java object properties from LotusScript programs. This functionality is known as LS2J.
LotusScript development	Good	Poor	WebSphere does not support LotusScript.
WebSphere MQ Integration	Excellent	Excellent	Both Domino and WebSphere have connectors and APIs to WebSphere MQ.
Serve HTML	Excellent	Excellent	Both Domino and WebSphere perform well as HTTP servers.
Multi platform	Excellent	Excellent	Both Domino and WebSphere have excellent histories supporting multiple platforms.
Enterprise Solution ready	Good	Good	Both Domino and WebSphere have connectors and APIs to various third party applications and technologies.

6

Scenario: JPA membership services

This chapter describes a simple scenario that illustrates how the Patterns for e-business can be applied to the creation of an IT solution. The scenario is a simplified version of an actual application created for a Japanese professional association (referred to here as the JPA) that wished to publish their existing and future member information on the Internet.

The goal of this chapter is to focus on the use of Patterns for e-business in relation to a specific scenario, not the implementation details of the recommended solution. The flow of this chapter first follows the Patterns for e-business recommended sequence of pattern identification. The remainder of the chapter addresses general performance considerations and security issues related to the recommended solution.

6.1 Overview

JPA has around 20,000 members. They currently manage their membership with a Lotus Domino application and Lotus Notes clients. Lotus Domino was used initially due to its workflow capabilities. The current environment is required because professionals in certain fields must be registered to practice in Japan. However, the information maintained is not currently used to support the individuals in achieving market exposure.

As part of the extension of the application to the Internet, JPA has decided to also provide the professional members with home page display capabilities and to allow them to self-administer their home pages and personal membership details.

In this chapter we describe the following steps for developing our sample scenario:

- Identify appropriate Business and Integration patterns, or Composite patterns that may map well to this scenario.
- Select Application patterns that apply.
- Select a Runtime pattern that satisfies business requirements.
- Review possible product mappings.
- Review guidelines and related information.

This is in accordance with the pattern for e-business Web site at:

`http://www.ibm.com/developerworks/patterns/`

However, to relieve you of any suspense, we can tell you up front that the resulting application consists of the following products:

- IBM Lotus Domino, used for data storage and updates.
- IBM DB2, used to contain indexed data from Lotus Domino.
- IBM WebSphere Application Server, used to process search requests.

6.2 Select a Pattern or a Custom design

In order to select an appropriate Pattern or Custom design, we need to develop a business scenario based on the high-level business goals for the solution. We then map each element of the scenario to the appropriate pattern.

6.2.1 Business goals for the solution

The first step toward developing our solution is to look at the business goals for the solution.

The primary goals for implementing this new system are the association's desire to:

- Make it easier for the public to find and contact their members through the Internet.
- Add value for their members by providing each member with their own home page with a virtual address that they can use on their own stationary and marketing material.
- Provide 24 hour access for the public to find members, rather than JPA office hours, as is currently the case.

The business drivers and the realities of the world have created the need for a number of key features that the new system will provide. Some of those features are:

- Enable the public to quickly and easily find contact details for the members from as many different device types as possible.
- Enable fast search response times for member searches within a 20,000 person membership, but also provide scalability up to 200,000 members.
- Provide a secure environment for members to update their individual home pages.
- Enable any of the following search criteria:
  - Last name
  - First name in combination with a last name
  - Business name
  - Country name and city name
  - Post code
  - Telephone, country and area codes
  - Specialty or professional segment
  - Number of years as a member of the association
- Ensure security and authentication of members' home page when updating.

As mentioned previously, a major consideration is the speed of the searches. Apart from that, there are a number of other points that we need to consider:

- We may need some special mechanism for indexing to speed searching, especially as membership levels rise to the expected 200,000 (or more) members.
- How can we update the index when a member's information is updated (whether by the member or by JPA staff)?
- How does the JPA staff access the membership information?

6.2.2 Business scenario

The business context of the system helps us to define and separate the participants in the system, which helps us to define which patterns to use in our solution to meet the JPA requirements.

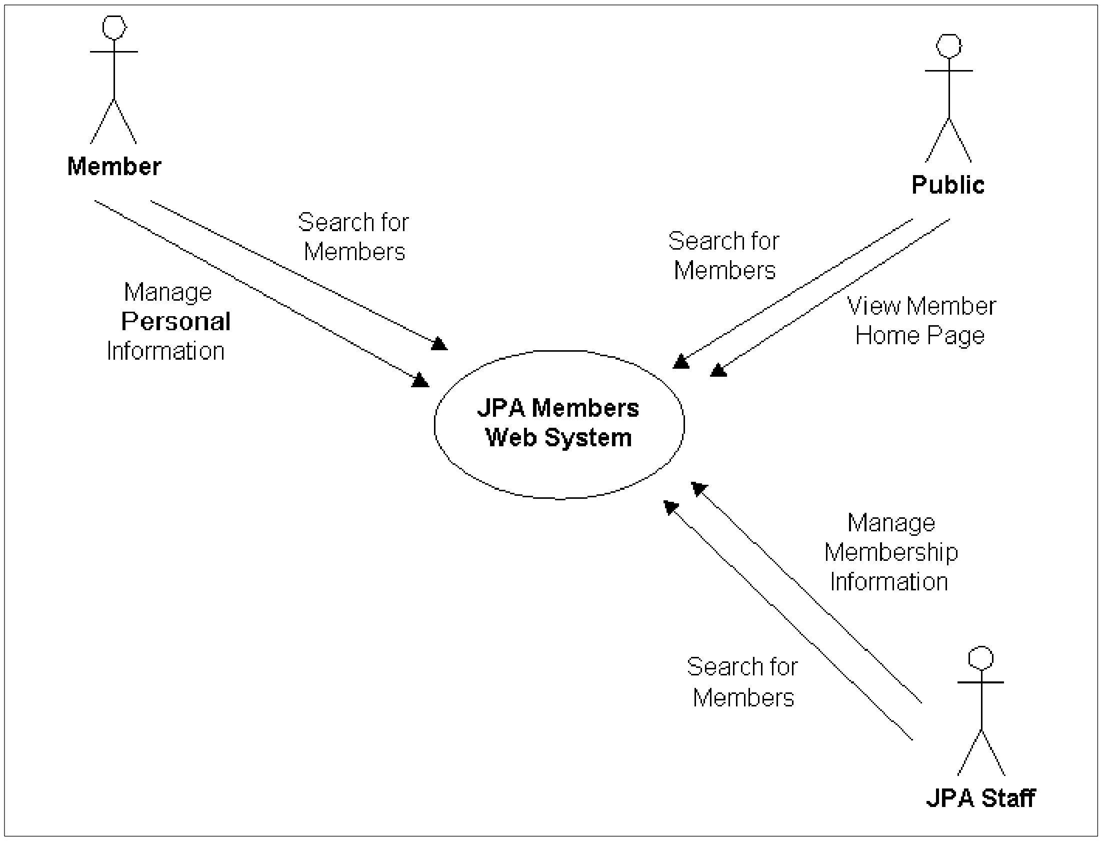


Figure 6-1 JPA business context diagram

The entities within the business context diagram are:

- Public

  Anyone, including members, who needs to search for members based on the search criteria specified previously.
- Member

  A member of the professional association who would access the system to update a subset of the personal information stored in the system. This could be information such as telephone numbers, addresses, and the rich text content within the page.
- JPA staff

  Association staff who need to manage the membership database and can access and modify the information about any member in the system. They would also be able to modify the same information that the member can modify (within his or her own record).

By looking at the business context diagram, we can identify the participants and their roles in the system. This allows us to identify the atomic e-business patterns that could be used in developing the solution.

6.2.3 Pattern selection

An examination of the business context just described and the roles that different entities serve in the context of the system provides information that can be used to determine the Business patterns that apply to this scenario.

JPA Membership Services is a fairly simple scenario, which requires only one obvious Business pattern: Self-Service. As explained by the authors in *Patterns for e-business: A Strategy for Reuse*:

> "The Self-Service business pattern captures the essence of direct interactions between interested parties and a business. These interactions can range from a simple information lookup to the execution of a complex business process." (p. 79)

Based on the requirement that public users be able to search and view membership information from a variety of devices through the Internet, an additional Business pattern is identified: access integration. This pattern is indicated when consistent and seamless access to an application using different mechanisms, including mobile devices and PDAs, is required.

Table 6-1 enumerates each entity found in the business context diagram, along with role and common task descriptions and associated Business patterns.

Table 6-1 Entity roles, tasks, and patterns

Entity	Role	Task	Pattern
Public	Inquirer	Search for member	Self-Service Access Integration
	Inquirer	View member home page	Self-Service Access Integration
Member	Inquirer	Search for member	Self-Service
	Inquirer	View member home page	Self-Service
	Updater	Modify membership record	Self-Service
JPA Staff	Inquirer	Search for member	Self-Service
	Inquirer	View member home page	Self-Service
	Manager	Create member records Delete member records Modify member records	Self-Service Self-Service Self-Service

Note that based on this information, other common Business patterns, such as Information Aggregation, Extended Enterprise, and Application Integration were not required.

Figure 6-2 represents the business context diagram provided in Figure 6-1, modified to include the reach of the Self-Service pattern.

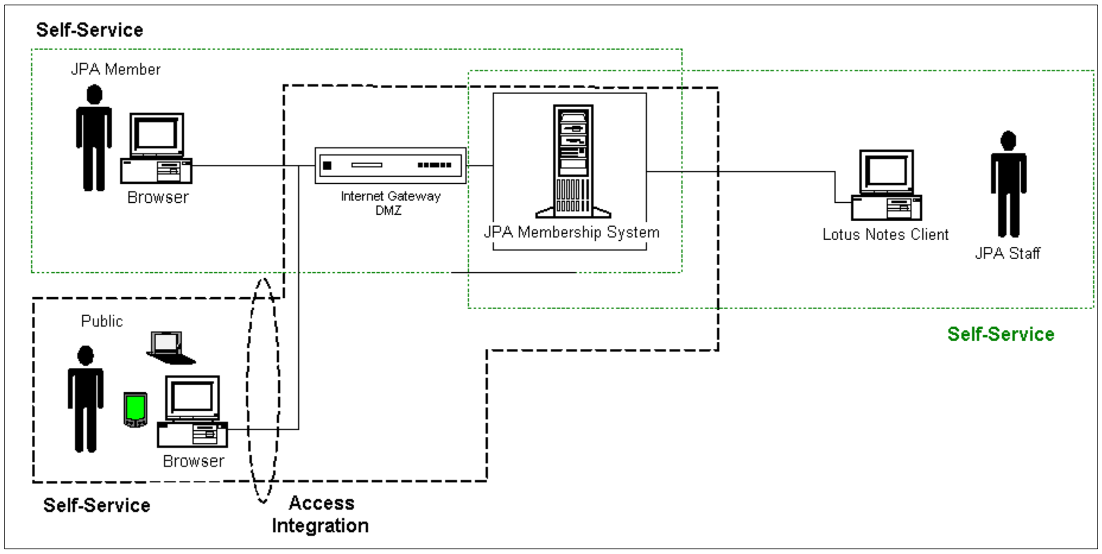


Figure 6-2 System context diagram with Business patterns

6.3 Select an Application pattern

Now that we have identified the Business patterns in this scenario, we need to identify the high-level logical components of the solution and their interactions. This will allow us to choose an appropriate Application pattern.

On a larger or more complex systems, you would probably use systems analysis and design tools such as Entity Relationship and dataflow diagrams. Due to the simplicity of our system we do not need to do a lot of analysis in order to understand and map the existing systems or to design our new system.

Before selecting appropriate Application patterns, we first analyze the processes that are required in the new system.

6.3.1 Process analysis

The process flow is illustrated in Figure 6-3 on page 91.

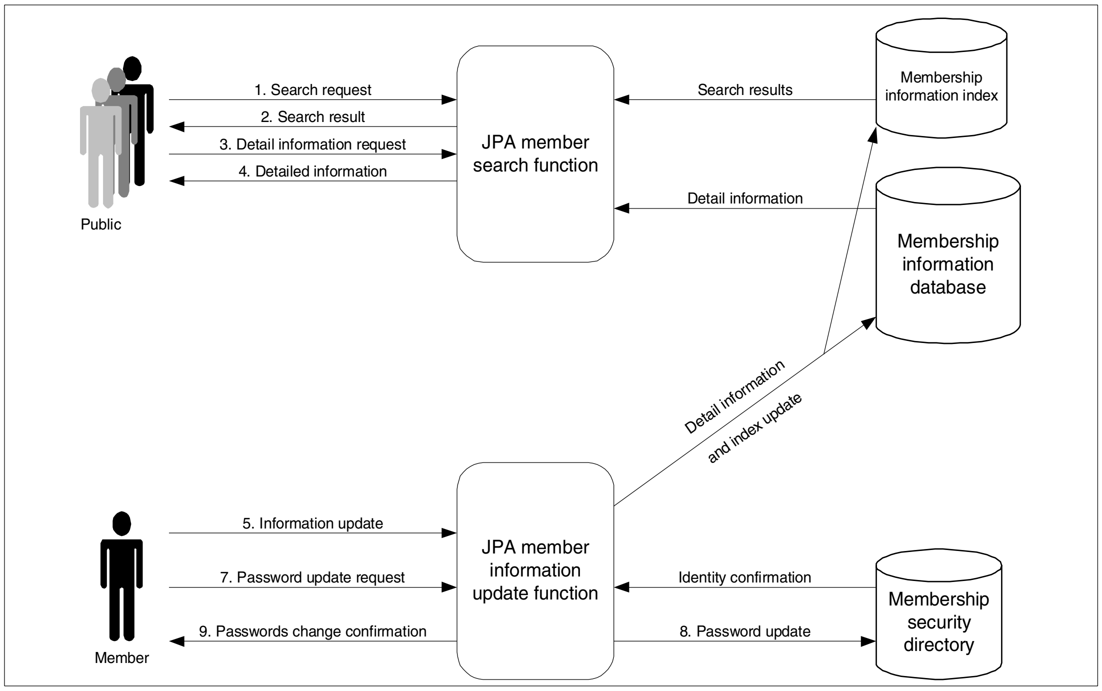

Public
1. Search request
2. Search result
3. Detail information request
4. Detailed information
JPA member search function
Search results
Detail information
Membership information index
Membership information database
Detail information and index update
Member
5. Information update
7. Password update request
9. Passwords change confirmation
JPA member information update function
Identity confirmation
8. Password update
Membership security directory

Figure 6-3 New consolidated process flow

The following sections outline the process flows identified in the Figure 6-3. Step numbers in the diagram correspond with the steps identified in these sections.

Searching for a member from an Internet browser

The following steps describe the new process of searching for a member from an Internet browser:

1. A member of the public fills in a Web form at the JPA Web site, requesting the JPA's membership directory be searched with one or more search conditions applied.
2. The Web site responds with the search results for the specified condition.
3. If the user wants to see the professional member's detailed information, the user clicks the link of the professional member name on the result screen.
4. The detailed professional member information is displayed on the Web browser.

Applying an update to a JPA member's home page

The following steps describe the new process of applying an update to a JPA member's own home page:

5. The JPA member is able to update their individual home page and some of the information that is stored within the database after the system has verified the user ID and password with the membership security directory.
6. When the member record is updated, the updated information is stored in the membership database and the search index is updated.

Changing the members' passwords

The following steps describe the new process of changing the members' passwords in the membership security directory:

7. The member fills in a Web form requesting the password change with the user ID and current password from the JPA Web site.

8. The new password is stored in the membership security directory after verifying the user ID and current password.
9. A password change confirmation is posted to the member's Web browser to confirm that the password change took effect.

6.3.2 Determining an appropriate Application pattern

Thus far, we have identified two Business patterns that apply to JPA Membership Services: Self-Service and Access Integration. Based on this analysis, and additional studies of published IBM Patterns for e-business, the "Account Access" composite pattern, which includes these two Business patterns, seems like an appropriate pattern to apply to this scenario. As described by Adams, et al., account access solutions provide customers with around-the-clock access to their account information (p. 71).

The Account Access composite pattern includes two mandatory pattern components: access integration and Self-Service. Both of these patterns apply directly to this scenario:

- Access Integration

  Provides for access to enterprise information through several devices, including regular browsers, PDAs, and mobile phones. JPA would like the public to be able to access JPA member information from a variety of devices.

  More specifically, the "Pervasive Device Access" application pattern, described in *IBM Patterns for e-business*, facilitates the extension of an individual application from browsers to pervasive devices, including PDAs and mobile phones.

- Self-Service

  Provides access to information stored in core business systems and databases. In this scenario, JPA would like its members to be able to access their own account information at any time. Public users must also be provided with access to JPA member information.

  The "Stand-Alone Single Channel" application pattern, described in *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, et al., narrows the focus of the Self-Service pattern for situations that do not require a structure for integration to multiple back-end systems. The JPA Membership Services system does not include any requirement to extend the reach of application data into other enterprise systems.

Figure 6-4 represents the patterns involved in this scenario, in the context of the Account Access composite pattern.

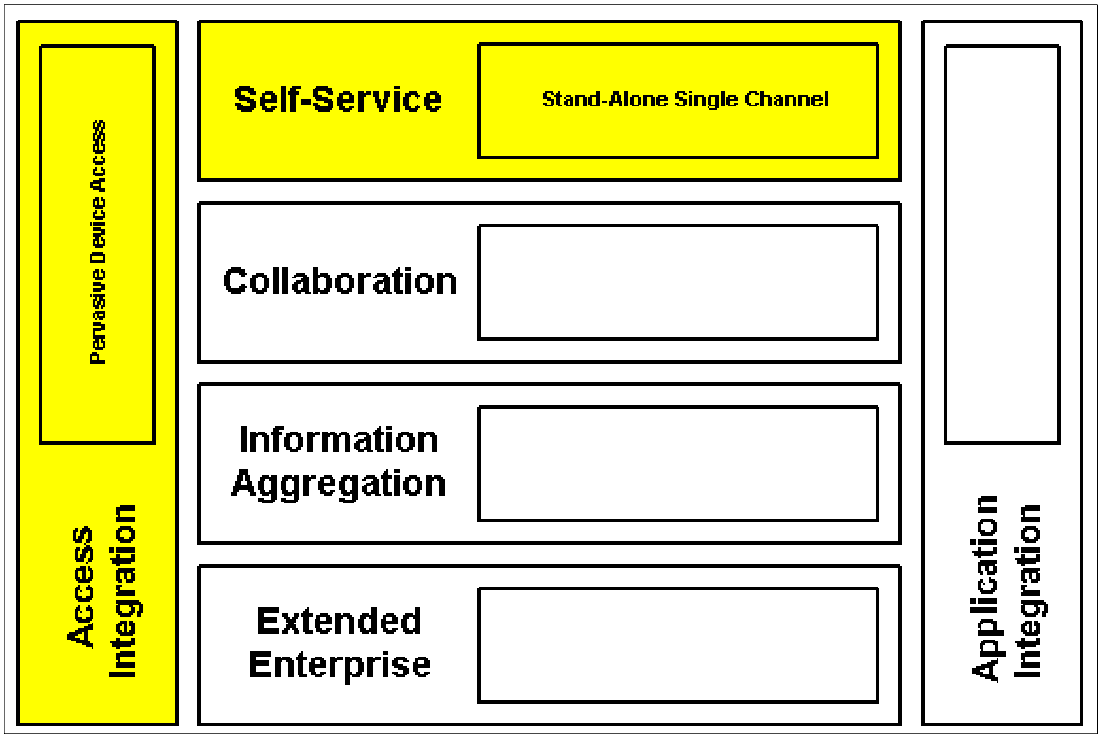


Figure 6-4 Account Access composite pattern applied to the JPA Membership scenario

In our case, the pervasive device access pattern applies almost directly to the JPA scenario, as shown in Figure 6-5, and will thus be mapped to our Runtime topologies described later in this chapter.

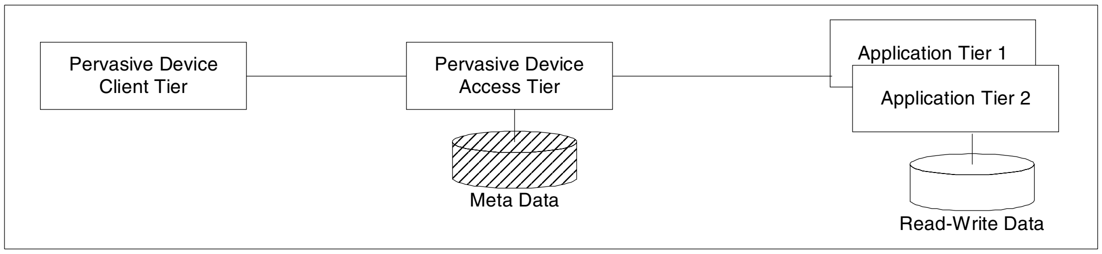


Figure 6-5 Pervasive Device Access pattern

Account access solutions may consist of additional components that are not yet included as requirements for the JPA system. These components include:

- The solution may optionally include an Information Aggregation pattern in cases where information from multiple accounts is summarized to provide a single unified portfolio view to the customer.
- The solution can also include the Collaboration business pattern as functions such as online chat with a customer service representative and help desk support are added to it.
- If the solution has any one of the optional Business patterns, the solution may optionally include an Application Integration pattern to seamlessly combine multiple Business patterns.

6.4 Review Runtime pattern

Now that we have selected an appropriate Composite Application pattern, account access, we can move one step deeper into our analysis by examining a Runtime pattern that applies to this scenario.

Runtime patterns are used to define the logical middleware structure supporting the Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes. These patterns also address how the processing logic and data are placed on these nodes. Like Application patterns, Runtime patterns represent logical constructs put together in a predefined structure that enables the solution to provide acceptable service levels to its users. The logical nodes can then be translated into a physical implementation, using vendor product mappings, recommended best practices, and personal experience. In other words, Runtime patterns describe the logical architecture required to implement an Application pattern.

The Runtime pattern selected for this scenario must take into account searching performance, scalability, and security.

6.4.1 Node types

The node types that will form this system are:

- A collaborative server
- A Web application server
- A transcoding proxy server
- A database server
- A directory and security server
- A firewall system

These nodes will be combined to create the final system solution, and will later be mapped to products.

6.4.2 Runtime diagram

In choosing a runtime topology that adequately meets the needs of the JPA Membership service, there are several issues to consider. Keeping in mind our choice of account access as our Composite Application pattern, and our underlying requirement for a scalable and secure system based on business requirements already provided, we can iterate through the Runtime topologies described in Chapter 3, “Lotus Domino and WebSphere Application Server patterns” on page 31 to select the best Runtime pattern to suite our needs.

- Domino-WebSphere single server

  This runtime topology does not provide the scalability that is required for the JPA Membership scenario, being confined to a single server.

- Domino-WebSphere multiple server

  This pattern assumes the use of Domino as the primary mechanism for providing content to the end-user, and there is not an inherent mechanism in this topology to scale the Domino server as required. For the most part, this describes the JPA scenario. The most significant issue with the selection of this pattern is in the lack of inherent scalability in the presentation layer, based purely on the components provided in the pattern. If there were far fewer public members accessing the system, this pattern might be a good fit.

- Web Redirector with Domino and WebSphere

  This pattern is an excellent fit for this scenario for several reasons. First, it includes both a collaboration server (Lotus Domino) and a Web application server (WebSphere). Second, by introducing a Web Redirector, caching and additional scalability features can be easily added. Third, to provide for a high degree of security, the DMZ does not contain any persistent data.

- WebSphere Application Server with Lotus Domino Services

  Since the JPA scenario assumes the use of Lotus Domino as the central repository for JPA membership data, and the JPA already uses Lotus Notes and Domino in its environment, this pattern, which assumes the use of a WebSphere J2EE application, is probably not the correct choice. If the JPA required additional integration with other back-end systems, or if the JPA did not already have a Lotus Notes/Domino solution that was being extended to the Internet, this pattern might be more seriously considered.

Based on this analysis, the pattern described in 3.2.3, "Web redirector with Domino and WebSphere Application Server" on page 41 is selected. In this scenario, we are also using an external directory server, which models the pattern described in 4.1.1, "Using an external directory for user information and authentication" on page 58.

This Hybrid Runtime pattern provides for:

- A DMZ that does not contain any persistent data.
- The ability to expand as JPA membership roles grow, thanks to the selection of products that scale well at each logical node,

Figure 6-6 illustrates the runtime topology that will be used in this solution.

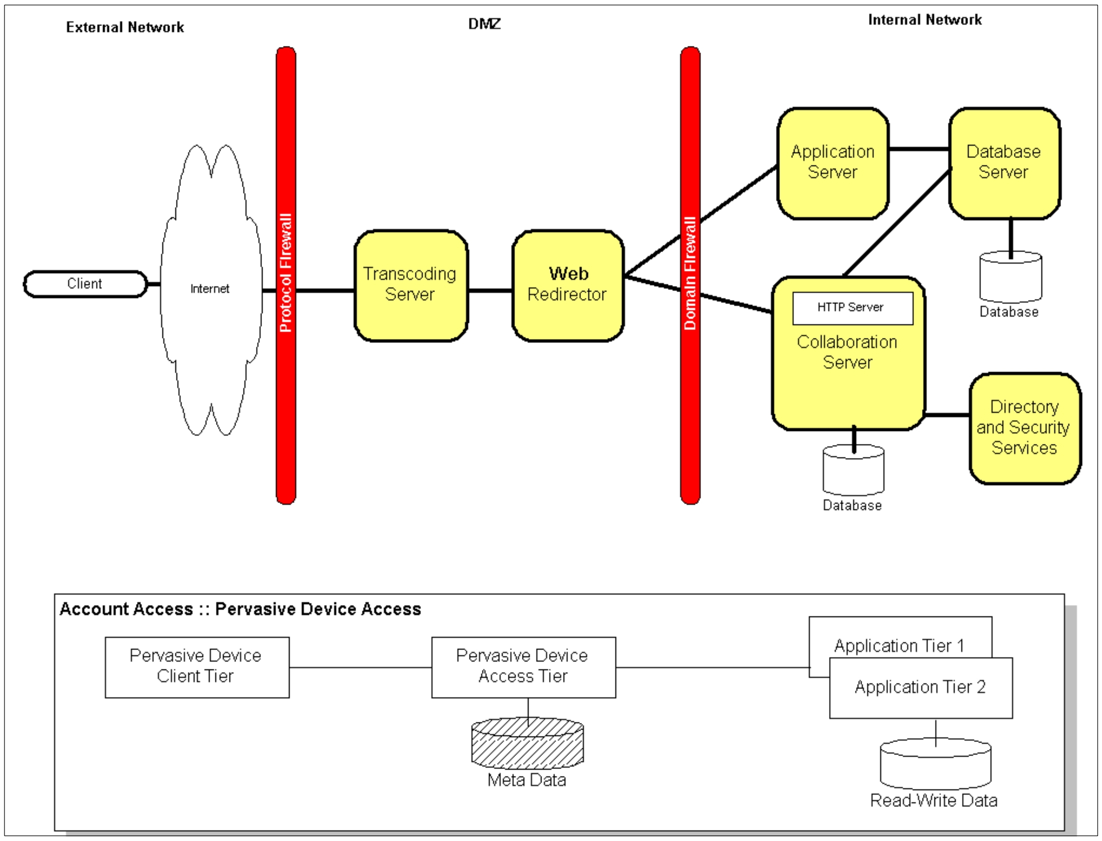


Figure 6-6 JPA runtime topology

The features of this Runtime pattern include:

- The only servers in the DMZ are the Transcoding Server and Web Redirector.
- All persistent data storage is in the internal network, protected by two layers of firewalls:
  - Protocol Firewall, blocking all ports to the DMZ with the exception of port 80 (HTTP) and port 443 (HTTPS).
  - Domain Firewall, blocking all traffic to the internal network except for traffic originating in the DMZ.
- The Web Redirector redirects client requests to either the Collaboration Server or the Application Server based on the composition of the URL.
- The Application Server is responsible for only the search facility.
- The Collaboration Server is used to provide virtually all presentation logic in the system.
- The Directory Server is used to authenticate all users attempting to access the Collaboration Server.
- The Transcoding Server is used to convert (transcode) returned HTML responses from the Collaboration or Application server based on the client device in use.

6.5 Review product mappings

In this scenario, the professional association plans to make 20,000 existing membership records available to the public via the Internet, and provide single home page service for JPA members, with each member being able to update their own record and home page. Eventually, there will be more than 200,000 membership records.

These business goals, stated explicitly in 6.2.1, “Business goals for the solution” on page 86, map to several products based on known product capabilities.

- Lotus Notes

  Lotus Notes clients are already used internally by JPA staff members to perform system maintenance and to update membership information. This ability will be retained in the new system.
- Lotus Domino

  Lotus Domino will provide all static and dynamic Web pages, and will also store the individual home pages for all members. Members are able to directly modify fields within their own membership document. The form will include a rich text field, allowing members to modify the content of their home page.
- WebSphere Application Server

  WebSphere runs a member searching servlet and returns the result as a JSP from a DB2 UDB data index table. The JSP will access data in DB2 through Java DataBase Connectivity (JDBC).
- DB2 Universal DataBase

  DB2 holds a data index for enabling quick searching of all membership records. It will be updated based on document saves in the membership database in Domino.
- IBM HTTP Server powered by Apache

  This server will be used as a base server for both the Web application server and the directory and security server.

- IBM Directory Server

  This server will provide an LDAP authentication service for access to edit member home pages. This will apply for the members who want to update their information or home page.

- WebSphere Transcoding Publisher

  WebSphere Transcoding Publisher (WTP), a part of WebSphere Everyplace Access, will provide member access via a variety of devices, such as PDAs, cell phones, and non-standard Web browsers.

Figure 6-7 provides a mapping of products selected in the JPA Membership scenario to the Runtime pattern identified in Figure 6-6.

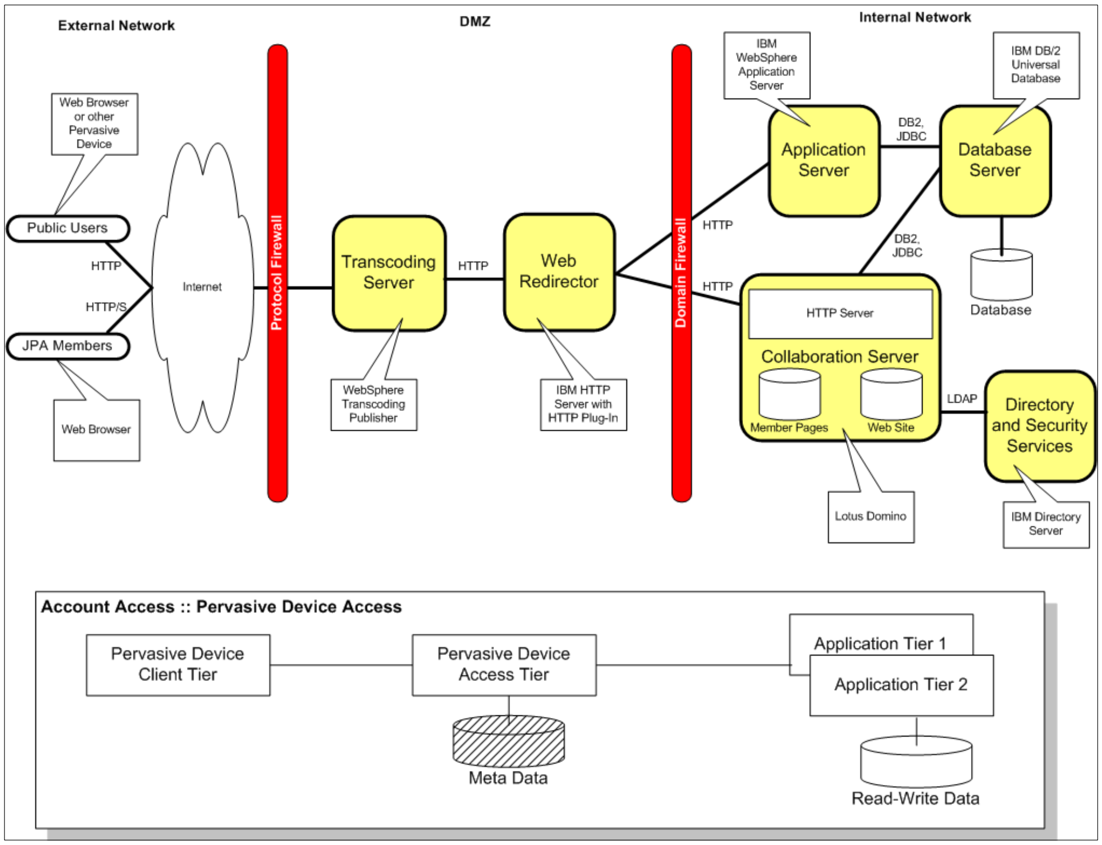


Figure 6-7 Product and protocol mappings

6.5.1 Collaboration server: Lotus Domino

Lotus Domino, which is currently used by JPA to internally manage JPA members, is the core of the new Web-based JPA system. Lotus Domino provides the following features in this solution:

- The home pages for each JPA member, and a method to edit the rich text content directly from a Web browser.
- A way to remap a friendly URL for each member. For example, rather than a member having a home page URL that looks like:

```
http://www.jpa.com/demo/JPAmSvcsDB.nsf/($All)/D03096FE634BFD5805256A1A000928A6?
OpenDocument
```

they would have a home page address like:

```
http://www.jpa.com/annettejones
```

Two databases are utilized in this solution. The first database contains members' home page documents. The only form within that database is the home page form; it has a LotusScript agent that runs on document save (WebQuerySave), which automatically updates the DB Index directly. Given the low number of updates required, LotusScript and the DB2 LSX should be adequate for the job. This database contains both presentation and data.

A second Domino database contains the remainder of the JPA Web site. Among other standard pages and images, it contains a search form which allows browser clients to enter search criteria and initiate a search. This form contains field validation for the fields from both JavaScript (client-side) and LotusScript (Java). This database contains both presentation and data.

6.5.2 Application server: IBM WebSphere Application Server

WebSphere provides us with a way to execute more than just lookups on the DB2 data. By programming the Java servlet that actually recalls the data from DB2, we can add significant functionality while still maintaining the speed and scalability of the system. At its present size, we could implement the whole system without WebSphere at all. If we were to do that, it would be at the expense of both speed and scalability (remember that the association intends to grow toward up to 200,000 members).

WebSphere is well suited to separating the presentation from the application logic.

6.5.3 Database server: IBM DB2 Universal Database

IBM DB2 Universal Database (DB2) will be used to provide the database indexes of the Notes membership database. It will provide scalability and speed so that searches of the membership return rapidly and accurately. Other relational databases could be used in this role, but we choose to use DB2 because:

- It can provide reliable and proven scalability.
- It is multi-platformed for versatility and horizontal scaling.
- Both WebSphere and Domino provide tools specifically for DB2.

6.5.4 Web Redirector: IBM HTTP Server powered by Apache (IHS)

The IBM HTTP Server powered by Apache (IHS) serves as the Web Redirector in this scenario. IHS provides excellent throughput and speed. It is well suited to its role in this environment. The WebSphere HTTP plug-in is used to redirect requests to either WebSphere Application Server or Lotus Domino.

The Lotus Domino HTTP task is used to provide access to most of the member home pages. The search servlet and JSP are the only application resources contained in WebSphere.

6.5.5 Directory and Security server: IBM Directory Server

IBM Directory Server provides an LDAP directory service that we can integrate with Domino to authenticate members when they need to update their personal information or their home page.

6.5.6 Transcoding server: IBM WebSphere Transcoding Publisher

The WebSphere Transcoding Publisher (WTP), a component of WebSphere Everyplace Access, provides access to the JPA Member data from multiple devices, including PDAs, cell phones, and other browser mechanisms.

WTP provides four different deployment options:

- As a network proxy
- As a reverse proxy
- As a filter in WebSphere Application Server
- As a plug-in in WebSphere Edge Server Caching Proxy (formerly called Web Traffic Express)

The deployment option recommended in this solution is *reverse proxy.* In this configuration, Transcoding Publisher acts as a proxy on behalf of the Web servers rather than on behalf of clients, as in the network proxy option.

Tip: A detailed discussion of IBM WebSphere Transcoding Publisher is beyond the scope of this chapter. For more information, see:

```
http://www.ibm.com/software/webservers/transcoding
```

The redbook *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5: Extending Web Applications to the Pervasive World*, SG24-6233, is also an excellent guide to using this product.

6.5.7 Platform choice

The key considerations in deciding what platform to choose are:

- Software availability for the platform
- Scalability of the platform
- Robustness of the platform
- Security of the platform

When we consider these factors, there are a large number of platforms that we have available to us.

Our selected Web application server and collaboration server (IBM WebSphere Application Server and Lotus Domino, respectively) are both available on platforms ranging from Windows 2000 Server and Linux microcomputers all the way through to z/OS (OS/390) mainframes. Our options are therefore not restricted by software availability for individual platforms.

Exactly the same can be said about *vertical* scalability, which can be achieved by implementing the system on more powerful hardware. The addition of *horizontal* scaling tools, such as WebSphere Edge Server, Domino clustering, and WebSphere cloning, ensures good scalability when demand starts to push the limits of the existing topology.

Platform robustness as a general rule increases as the platform selection moves from Windows 2000 Server through to mini, mid-range, and mainframe computers. The selection of which actual platform to use will be dependent on the association's requirements for system availability.

While each of the servers in our environment could be installed on different platforms, from an administrative perspective we recommend selecting one platform and using that for all of the servers. That way, systems administrators do not need to be multi-skilled in order to manage the various servers.

6.6 Review guidelines and related links

This section contains information about:

- General design guidelines for two of the most critical use cases of the JPA system.
- High-level performance guidelines related to some of the more important nodes in the suggested Runtime pattern.
- High-level system management guidelines for the JPA Web site.

Note that this chapter does not include full details related to the development of the JPA system, which is an exercise left up to the reader. The focus of this chapter, and this section, is to simply guide the reader through the activities and processes that would normally begin after the analysis that has taken place in prior sections.

See the patterns web site at `http://www.ibm.com/developerworks/patterns` for more design, development, and management guidelines that should be considered when implementing a solution based on the Account Access composite pattern.

6.6.1 Design guidelines

The use cases of interest in the JPA system that require further exploration are:

- Updating JPA member information
- Searching the JPA system

These are the two use cases that will consume most application development time and consideration during the implementation of this system, and they are the most critical to the success of the new Web site. These use cases were described in 6.3.1, “Process analysis” on page 90, and illustrated in Figure 6-8. Note that for the sake of simplicity and focus on the two use cases being discussed, the Directory Server component and Transcoding Server are not included in this figure. The Transcoding Server would act simply as another logical node just before the Web Redirector, but all results, including Search Results (JSP) as indicated in the figure, would funnel through the transcoder to properly format the returned HTML based on the client’s access device.

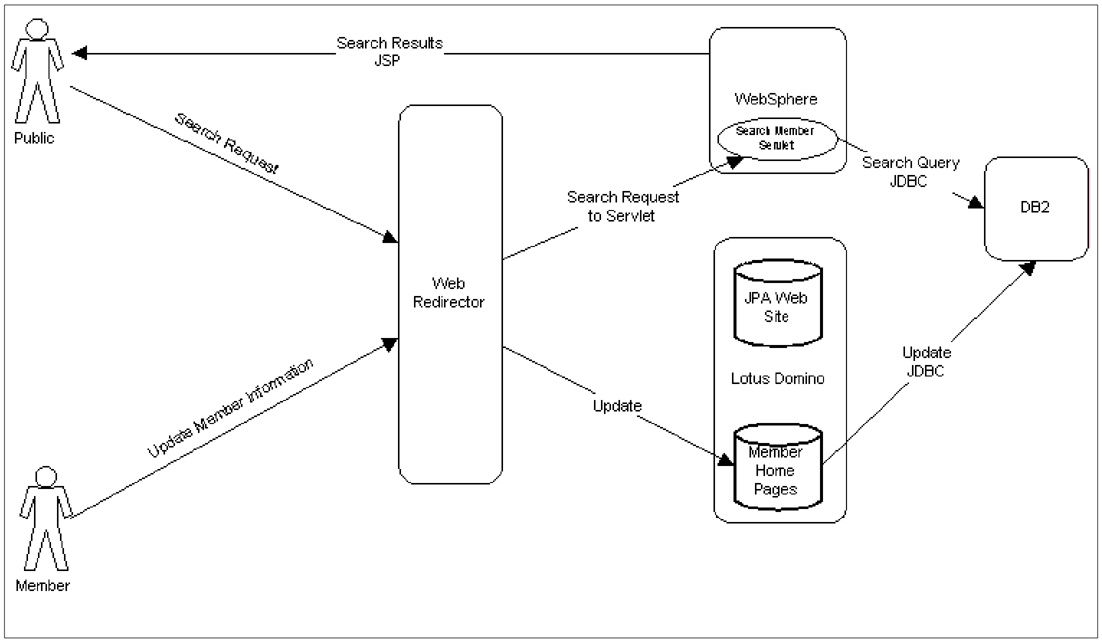


Figure 6-8 Information flow for two use cases: Member updates and public searches

Updating and creating JPA member information

One of the key business goals of this system is to give JPA Members the ability to modify their own member information. The JPA Member information will be modified in the Lotus Domino Home Pages database, then synchronized with IBM DB2 to handle search requests from public users.

1. The JPA member authenticates with the JPA member database through an encrypted (SSL) login.
2. The user clicks on the Update Home Page link, which locates the authenticated user's home page document in the Lotus Domino Home Pages database.
3. The user is able to change information contained in their home page, and submits this information back to Lotus Domino.
4. When the document is saved back to Lotus Domino, a Java save agent is initiated in Lotus Domino that uses a JDBC connection to write all home page information that may be included in search criteria to the IBM DB2 database. Alternatively, the IBM DB2 LSX or DB2 ODBC driver might be used in LotusScript.

Note that the steps required to create a new home page entry for a new JPA members would all be performed by JPA staff, since there is not a business requirement for "self-registration" for new JPA members. Since JPA staff will use their Lotus Notes clients to access the database, they would simply create a new JPA member document.

Searching JPA member database

In order to achieve the business objective of scaling up to potentially 200,000 (or more) members, and allowing the public to search the JPA membership database using the criteria described, we will allow the public to perform searches through WebSphere Application Server against the IBM DB2 database.

The use case creating or updating membership, described previously, is responsible for populating information into the DB2 database that will ultimately be searched. The goal of this use case is to allow the user to enter search criteria, to initiate the search, and to return search results.

1. From the home page, the user clicks a “Search JPA Database” link. The Transcoding Server passes this request to the Web Redirector, which then passes the request to the Lotus Domino server to be processed.
2. The search page is returned from the Lotus Domino server through the Transcoding Server, which reformats the contents of the page based on the device that the user is using. This page includes the following fields (which were identified as search criteria during the establishment of the original business goals):
   – Last name
   – First name in combination with a last name
   – Business name
   – Country name and city name
   – Post code
   – Telephone, country and area codes
   – Specialty or professional segment
   – Number of years as a member of the association
3. The submit on the user form will perform a post to a servlet in WebSphere Application Server.
4. The search is performed in the servlet against the IBM DB2 database.
5. A JSP page is returned with all search results, again through the Transcoding Server, which formats search results based on the client device in use.

6.6.2 Performance guidelines

The runtime topology selected will support future membership growth through the use of DB2 to facilitate fast membership searches, which is currently the only burdensome task initiated by an end-user (public member). Due to the nature of this search, which may be initiated by any public user, the Application Server and Database Server nodes of the solution will be solely responsible for this task.

To fully consider system performance, we need to examine all of the server platforms and software applications that are a part of the solution. This section describe the following nodes in our selected Runtime:

- Web Redirector (IBM HTTP Server)
- Application server (WebSphere Application Server)
- Collaboration server (Lotus Domino)

The information in this section focuses on performance considerations with the JPA scenario specifically, along with some additional product information. For general performance and scalability information, see Chapter 9, “Scalability and redundancy” on page 161.

Web Redirector (IBM HTTP Server) considerations

The IBM HTTP Server is powered by Apache, an open-source HTTP server that is available on numerous platforms. IBM has added several performance enhancements to the latest release of Apache HTTP server, including the Fast Response Cache Accelerator.

For more information on IHS and its Fast Response Cache Accelerator, see:

http://www-4.ibm.com/software/webservers/httpservers/about.html

While Microsoft Internet Information Server (IIS) was not included in the selection of products for this solution, it very well could be, since it is supported through the WebSphere HTTP plug-in architecture. For information regarding IHS performance in comparison with IIS, see:

http://www.ibm.com/software/webservers/appserv/wcat.pdf

The HTTP plug-in, used by the Web Redirector, adds an additional method of scalability by providing integrated load-balancing features. Of course, adding load-balancing features at this node would presume the existence of multiple application and collaboration servers from which to distribute client requests. More information about this plug-in technology can be found in 3.1, "WebSphere HTTP Server plug-in architecture" on page 32.

Another contributing factor to the performance of the Web servers is encryption. This is because encrypting connections or files can add significant load to the Web server's CPU. Security is discussed in 6.7, "Security considerations" on page 105. In our scenario, the public needs to be able to easily and quickly search for members. Since all of the information that is presented to the public is non-confidential in nature (otherwise the member would not allow the information to be visible), there is little point in securing those connections. This means that for the majority of the traffic, the connection will not be encrypted. When members view or update their personal information, that should be encrypted to ensure that information can not be accessed inappropriately.

SSL should only be enabled for the members viewing the extended information on their own page.

WebSphere performance considerations

In our scenario, the WebSphere Application Server performs, at least initially, only one function: providing search results through a JSP.

Additional information pertaining to general WebSphere performance considerations can be found in the following Redbooks:

- *DB2 UDB/WebSphere Performance Tuning*, SG24-6417
- *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher*, SG24-5858

Collaboration server (Lotus Domino) performance considerations

The Domino HTTP task will be used to service end-user page requests, through the Web Redirector. Even with the Web Redirector, performance of the Domino HTTP task must be tuned, since the redirector simply reroutes user requests through itself to either the collaboration server or the application server. For more information about HTTP task performance, and general Domino performance considerations, see:

http://www.lotus.com/home.nsf/welcome/performance

The connection between a member and the system, once they have logged in, is implemented via an SSL-enabled link. This introduces the added load required to support SSL encryption. The added load when SSL is enabled is significant. Fortunately for us, the association estimates that there will be fewer than 300 secure logins per day after the initial curiosity has worn off. While 300 secure connections per day is a reasonable number, that is spread over the whole day, so that the number of concurrent secure connections will be even lower, on the order of eight to nine concurrent sessions on average.

These figures are based on the assumption that members will not be logged in for more than one hour, members will only be accessing their own home pages during an 18 hour period per day (numbers for the remaining six hours being significantly lower than the other 18 hours), and that the number of concurrently logged in members is relatively constant for the 18 hour period.

This does not represent a significant load on the Domino server, but the actual numbers of concurrently logged in members should be reassessed once the system is in place to prevent SSL encryption from becoming a crippling force on the site. Should the load for the encrypted connections start to become significant, minor changes in the selected runtime topology might be made, such as:

- A second Domino server can be added, which replicates with the first any changes made to member documents. If this is implemented, you might also consider clustering those two Domino servers so that updates are immediately visible on the server that the public conducts their searches from. The HTTP plug-in, in the Web Redirector, may be configured to distribute load between the two servers. Optionally, another load-balancing mechanism, like Lotus Internet Cluster Manager (ICM), can be used.
- A second Domino server can be added that is used exclusively by JPA members, with their own new entry point from the Internet into the system. Changes made to member pages, like the previous suggestion, could be replicated with the primary Domino server.

Another point to consider is sizing the Domino server appropriately. A Domino server that only needs to handle requests from Notes clients will have significantly less load placed on it than a server which is serving the same number of Web browser clients. This is because, with a Notes client, the client does the rendering and interpretation of the design, but with a browser, the Domino server must do all of the work. Accordingly, Domino servers that predominately serve Web browsers should have more memory and faster CPUs than are necessary for a Notes client-centric Domino server.

For more information on optimizing Domino performance, see:

`http://www.lotus.com/ldd` (search the Documentation Library)

`http://www.lotus.com/developers/itcentralnew.nsf` (See the Performance Zone)

6.6.3 System management guidelines

Apart from normal maintenance of the membership by JPA staff, there should be very little maintenance needed on the system. While the goal of this chapter is to focus on the use of Patterns for e-business in the analysis and initial design of this scenario, we provide some general system management guidelines that pertain to the JPA Web site.

- Perform™ periodic rebuild of the membership data index for searching in DB2 from the Domino membership database.

  We do this both as a common housekeeping operation to make sure the DB2 search index is in sync with the actual Domino data, but also to find updates to the Domino data that have occurred while DB2 has been down or stopped. This is because we accept that a member can update his or her own home page and save it into the Domino membership database even if the connection between Domino and DB2 is down or DB2 is stopped. To synchronize membership data in Domino and its index for searching in DB2 after the connection or DB2 comes up, we can make a scheduled agent in Domino to initiate an update index process in WebSphere. Alternatively, Lotus Enterprise Integrator can be used to perform this task on a scheduled basis.
- Standard Notes administrative tasks (fixup, updall, and so on).
- Occasional server reboots (depending on the ultimate operating system choices for the production system).
- Regular backups of both the Notes databases and DB2 index databases.

For ongoing management of the environment, the following items should be considered:

- Specialist tools which monitor Simple Network Management Protocol (SNMP) traps may be appropriate.
- Analysis of the Domino, Domino Web, and WebSphere logs on a regular basis, preferably daily.
- Regular security testing and analysis.
- Regular testing of the system to ensure it is still operating as expected.
- Regular testing of the backups to ensure that a restore from backup, if needed, would be successful.

6.7 Security considerations

In order to proactively deal with the present and emerging threats and vulnerabilities to business assets over the Internet and intranets, security needs a very special focus right from the beginning of any design process. If this focus is not taken, it is possible that you might need to do a complete redesign to retrofit the necessary security measures. In this section we discuss how the different security measures are applied and utilized in the JPA membership scenario. For more information you can also refer to the Redbooks *Enterprise Security Architecture using IBM Tivoli Security Solutions,* SG24-6014 and *IBM WebSphere V5.0 Security WebSphere Handbook series*, SG24-6573.

6.7.1 Basic security requirements

The security needs for any business are traditionally based on risk assessment and management. How risks are managed is a business decision based on the business's assessment of the risks involved in not providing various security measures as compared to the benefit achieved by those measures. The factors that influence the decision to choose a particular security architecture depend mainly on the type of application the business is building and the business value of the transactions and data that the application will support. The security requirements for a business generally include:

- Access control
- Flow control
- Audit control
- Credential management
- Integrity

To learn more about a security methodology based on these categories, see "End-to-End Security: A Method for Designing Secure Solutions" in *IBM Systems Journal, Volume 40, No. 3, 2001,* available at:

http://www.research.ibm.com/journal/sj40-3.html

In the following sections we describe how the topology we chose for this solution addresses the various business security needs. To do so, we start directly with the runtime topology and product mapping.

6.7.2 Security analysis

When operational, this site will be open for all users over the Internet, so this scenario demands the maximum security possible to ensure that the organization's internal resources and data are not compromised.

Referring to Figure 6-6 on page 95, the Runtime model selected includes the following security features:

- Both application servers are placed behind the domain firewall, only accepting traffic originating in the DMZ.
- The Web Redirector and Transcoding Server are the only servers contained in the DMZ.
- The first firewall blocks all ports with the exception of port 80 (HTTP) and port 443 (HTTPS).
- The second firewall blocks all traffic that does not originate from within the DMZ. More specifically, only traffic from the Web Redirector is allowed through the domain firewall.

In the case of this scenario, it is not felt that additional firewalls are required, and that the standard two-firewall scenario is more than adequate to address the security needs of the JPA.

6.7.3 Security requirements for this scenario

In this section we discuss how our topology addresses the various security requirements.

Audit

All Web accesses are logged at the Web redirector level. Domino serves all core Web site content, as well as member pages. In order to ensure that the data in Domino is not compromised, all successful and failed login and access attempts to Domino resources are logged. The firewalls restrict the traffic that goes through them to ensure that configured sources are the only ones that can get past them.

As search requests do not pose a security threat, and searches may be requested by anonymous system users, only search failures (connectivity problems, for example) will be logged.

Access control

In this scenario, the three user types are provided with access to Lotus Domino resources as follows:

- Public Users

  Public users are provided with read-only access to all Lotus Domino databases. Public users do not require authentication.

- JPA members

  Since JPA members will be provided with the ability to modify their own Web page, an Authors field in the JPA Membership database will be used to restrict access, containing the authenticated name of the JPA Member primary user.

- JPA staff

  JPA staff should be able to modify any documents in the database, including not only the JPA Web site, but also all JPA membership pages. This will allow JPA staff to correct problems that members may have with their Web sites, or assist members as required with home page modification. Therefore, Authors fields in the Member database will not only include JPA Member names, but will also include the name "JPA Staff".

Following these rules, the Access Control List for the JPA Web Site database should be defined as in Table 6-2.

Table 6-2 Access Control List for JPA Web Site database

Access Level	Who
Manager	JPA Domino Admin
Designer	Internal Domino Server
Editor	
Author	Members, Anonymous (both with all check boxes unchecked except create documents)
Reader	
Depositor	
No access	[default]

The Access Control List for the JPA members database should be set as shown in Table 6-3.

Table 6-3 Access Control List for JPA members database

Access Level	Who
Manager	JPA Domino Admin
Designer	Internal Domino Server
Editor	
Author	Author access - Members (with all check boxes unchecked except create documents)
Reader	Anonymous (with all check boxes unchecked)
Depositor	
No access	[default]

From a WebSphere security perspective, since the public is able to perform searches against DB2, the search servlet resource in WebSphere will allow any user to submit search requests through the search servlet.

From a relational database perspective, the only two entities that will access the DB2 database include the search servlet and Lotus Domino (on member information updates), both via JDBC. A single user ID will be created that the JDBC driver will utilize, providing read/write access to the DB2 database through JDBC calls.

Flow control

The Protocol and Domain firewalls, one on each side of the Web Redirector and Transcoding Server, provide a complete separation of the internal network and its data from the Internet. This prevents the possibility of hacking, or intentional or accidental damage to JPA membership data from the Internet.

Credential management

Anonymous users are allowed to search for and view member pages. JPA members who wish to edit their own information must authenticate with the directory and user security server. Lotus Domino challenges any user who attempts to check out to provide a user ID and password over SSL. The Directory and User Security Server, implemented with an LDAP directory, is used to control JPA Member authentication with the site.

Solution integrity

Since the member records contain personal information about the members, we must determine if it is desirable to encrypt the sensitive information fields within the members database. In addition, we must determine if the database itself should be encrypted, preventing back-door file access to the database. Both of these features have performance impacts on the Domino server. The degree of the performance penalty is dependent on:

- The number and size of fields in the member documents that are encrypted
- The degree of database encryption selected: simple, medium, or strong

Since the Domino server will be placed in the “Inside Network”, behind two firewalls, in addition to being protected through Domino’s own security system, there is very little chance that this data will be compromised.

Therefore, to increase overall system speed, no fields in the database will be encrypted, and the database itself will not be encrypted.

Like all security issues, these factors will need to be constantly reassessed once the system is in production so that changes in JPA policy, privacy laws, or new malicious risks do not open the JPA to criticism or legal action.

6.8 Summary

In this chapter we described the process of utilizing e-business patterns to analyze a given scenario, the extension of the JPA Membership Web site to the Internet. Starting with an investigation of JPA’s business requirements, the steps completed in this chapter followed the Patterns for e-business general process guidelines:

1. Identify Composite patterns

   In this scenario, the Account Access composite pattern was identified as a good match to JPA’s business requirements. This pattern is well suited to assist JPA in it’s requirements to provide members direct access to their accounts, and to provide support for pervasive devices.

2. Identify Business and Integration patterns

   The Account Access composite pattern assumes the use of access integration and Self-Service patterns.

3. Identify Application patterns

   The Application pattern tied to the Access Integration pattern that is best suited in this scenario is “Single In-Line.” In this scenario, the Pervasive Device Access application pattern, one of the Application patterns available under the Access Integration pattern umbrella, is the best fit. The Stand-Alone Single Channel application pattern, one of the Self-Service business patterns, is also recommended.

4. Identify Runtime patterns

   In this situation, we created a Runtime pattern based on a combination of the Pervasive Device Access application pattern and the Stand-Alone Single Channel application pattern. We also used Hybrid Runtime patterns as input.

5. Identify product mappings

   Based on business requirements related to scalability and the current JPA environment, products were mapped to each node in the developed Runtime pattern.

After the pattern selection was complete, we did a partial review of design, development and management guidelines for the application. We followed this with a discussion of performance and application security considerations.

7

Scenario: Laura’s Gadgets e-shop

This scenario describes a successful regional retailer, (Laura’s Gadgets) that wishes to provide customers with an online sales site and automate more of their supply chain functions. The company specializes in the sales of connected-ware, such as PDAs and mobile phones. The site should be integrated with existing back-end systems.

7.1 Overview

The business currently has several shops which sell mobile phones, PDAs, portable audio devices, and accessories. The current IT environment includes:

- A product catalog management system based on Domino.doc
- A stock inventory and supply application
- Bank-supplied terminals for processing credit card settlements

The new e-shop application should provide:

- Quick Web browser-based product searches
- Customer registration
- Shopping cart and order tracking for customers
- Integration with the existing catalog and inventory systems
- Web browser-based credit card settlements

The methodology we use in this chapter is based on the e-business patterns and processes described in Chapter 1, "Patterns for e-business" on page 1. As outlined in that chapter, the process follows these steps:

- Identify appropriate Business and Integration patterns, or Composite patterns that may map well to this scenario.
- Select Application patterns that apply.
- Select a Runtime pattern that satisfies business requirements.
- Review possible product mappings.
- Review guidelines and related information.

7.2 Select a Pattern or a Custom design

In order to select an appropriate Pattern or Custom design, we need to develop a business scenario based on the high-level business goals for the solution. We then map each element of the scenario to the appropriate pattern.

7.2.1 Business goals for the solution

The business goals for the solution include:

- Business growth
- Increase the number of items available
- Improve customer relationship management
- Reduce the cost per sale, in other words, maximize the profit per sale

Some of the key features of the system that will contribute to these goals are:

- Quick and easy access to the product catalog
- Online shopping through a Web browser
- Tracking of customer orders
- Secure input and storage for customer information such as name, billing and shipping addresses, credit card numbers, and perhaps other sensitive data

- Publishing of updates made to the existing Domino.Doc product catalog management system
- Automated order and fulfilment process beginning with customer order submission through a Web browser, and continuing through the order shipment
- Order acceptance and status information via e-mail
- Online access reports and sales reports

To provide these features, it is necessary to provide integration between the new Web application and the existing systems. The major integration points are:

- A transaction-based connection to the current credit card settlement system
- A transaction-based connection to the existing inventory and supply application
- A mechanism to publish and maintain a product catalog for the system based on the existing Domino.Doc product catalog maintenance system

7.2.2 Business scenario

Figure 7-1 is the business context diagram which represents Laura's Gadgets e-shop.

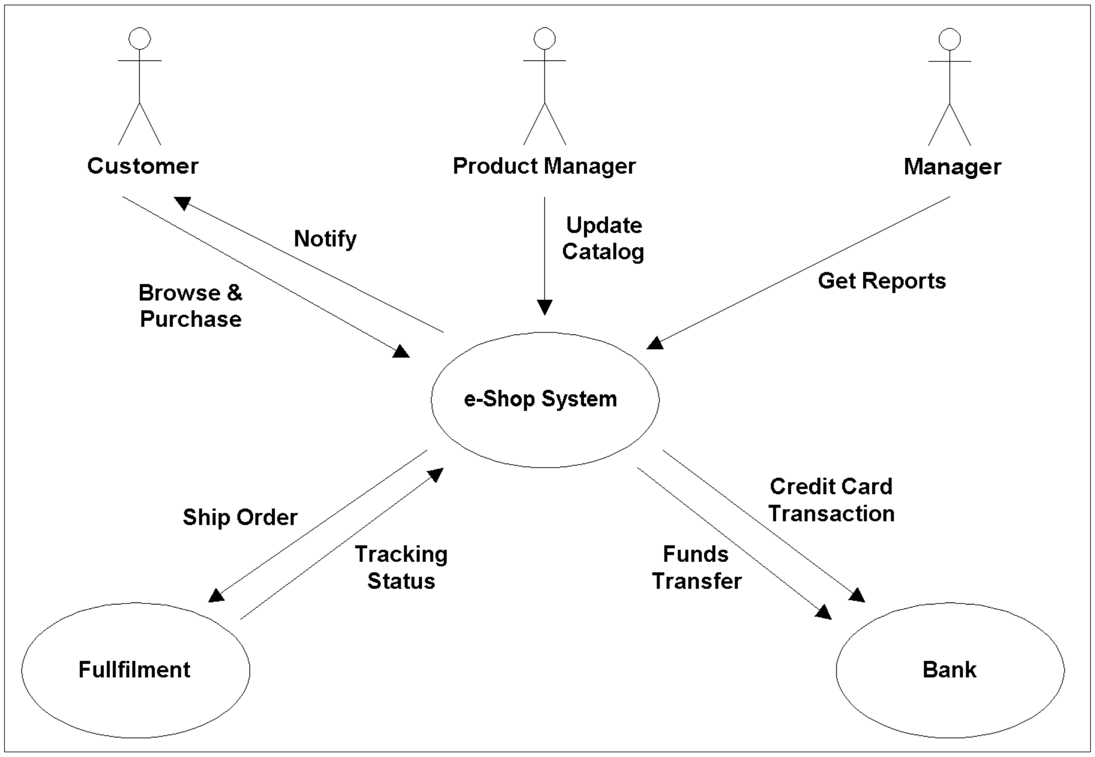


Figure 7-1 Business system context diagram

The business context diagram identifies the following entities:

- e-Shop System: The new e-shop application as seen by customers, managers, and product managers.
- Customer: Anyone who wants to browse the catalog or purchase items from Laura's Gadgets e-shop.
- Product Manager: The person responsible for managing products in the e-shop.
- Manager: The manager receives Web site reports about page hits and sales totals.

- Fulfillment: This system manages the delivery of the goods. It receives orders for shipping and tracks order delivery status.
- Bank: Handles credit card processing for customer purchases.

7.2.3 Pattern selection

Since different parts of the solution require different Business patterns, the solution will require a Custom design or a Composite pattern. The IBM Patterns for e-business Web site describes the *Electronic Commerce* composite pattern. A variation on this pattern is a perfect fit for the e-shop application.

The basic Electronic Commerce composite pattern is composed of the following Business patterns, as shown in Figure 7-2:

- A Self-Service business pattern that provides customers access to Web site functions such as browsing a catalog, placing an order, and making a payment. For the e-shop application, a Self-Service business pattern is also needed to provide the catalog maintenance functionality.
- An Information Aggregation pattern that aggregates information from multiple sources into a unified catalog of items. Our e-shop application's catalog is slightly simpler as it has only a single source for the catalog. An Information Aggregation pattern is also needed to provide the reports that are part of the business goals.
- An Application Integration pattern that combines the Self-Service pattern and the Information Aggregation pattern to provide a unified solution to the customer. In our e-shop application, the Application Integration pattern can be used to access the existing inventory, fulfillment, and catalog management systems.

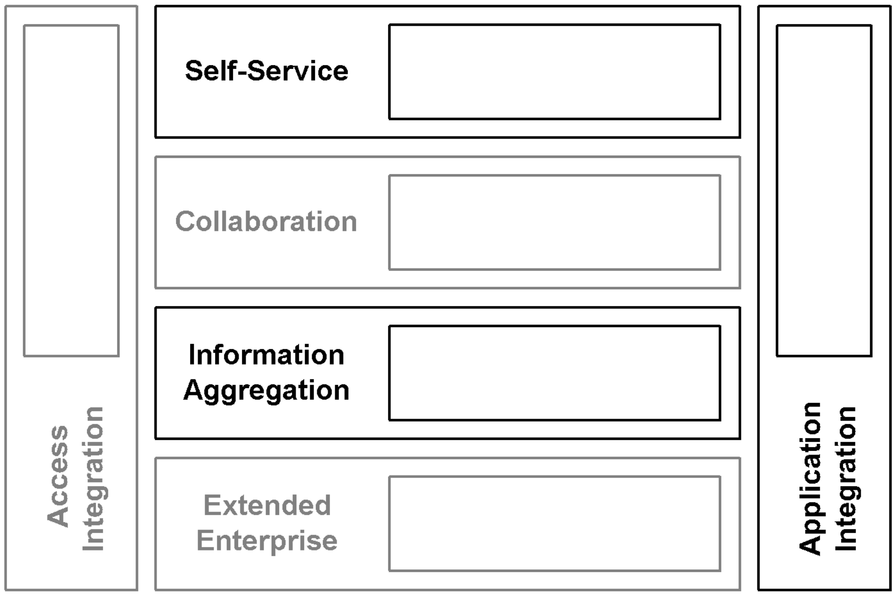


Figure 7-2 The Electronic Commerce composite pattern

There are also a few optional variants to the Electronic Commerce composite pattern:

- An Access Integration pattern that provides for more sophisticated options, such as personalization and pervasive device access, aimed at increasing the user-friendliness of the site

- A Collaboration business pattern that provides functions such as automatic order confirmation through e-mail or online chat capabilities with customer service representatives
- An Extended Enterprise pattern that can be used to implement a direct connection with the shipping company that delivers the order to the customer

Personalization and pervasive device access are not goals for the initial implementation of the application, so our solution does not require the optional Access Integration pattern at this time.

Our business goals do include a customer notification via e-mail of order shipment, so we need to include the Collaborative business pattern to cover this requirement.

Finally, we need an Extended Enterprise pattern to cover the interaction with the current credit card settlement system.

Figure 7-3 illustrates the Electronic Commerce composite pattern including the optional Business patterns needed to satisfy the e-shop application's business goals.

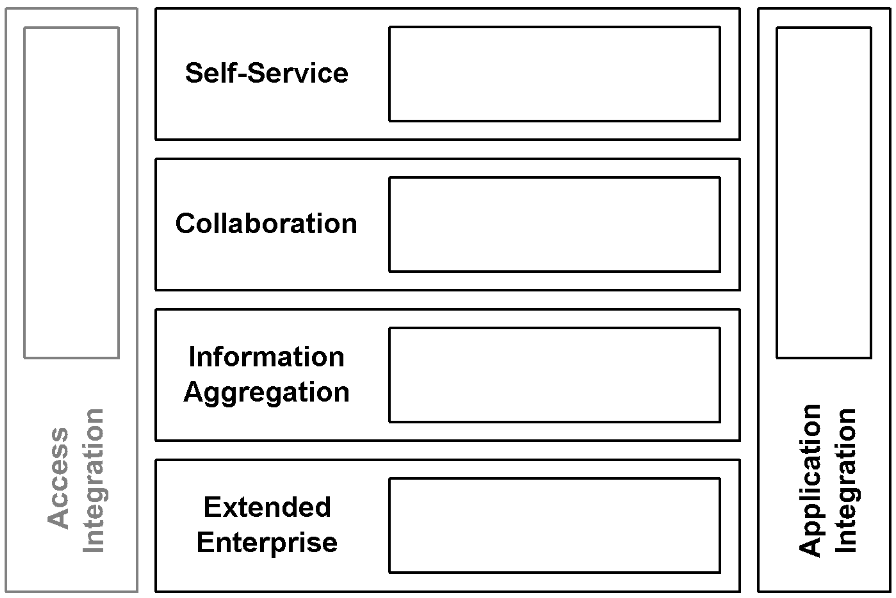


Figure 7-3 The Electronic Commerce composite pattern with variants

Table 7-1 summarizes the Business pattern selection analysis. It shows how each of the Business patterns that makes up our variation of the Electronic Commerce composite pattern applies to the entities and tasks in the business context diagram shown in Figure 7-1 on page 113.

Table 7-1 Business entities, tasks and patterns

Entity	Tasks	Pattern
Customer	Browse catalog Place items in shopping cart Register Submit and pay for order Browser order status	Self-Service
Customer	Receive shipping notification	Collaboration

Entity	Tasks	Pattern
Product Manager	Maintain product catalog	Self-Service
Manager	Browse e-shop reports	Information aggregation
Fulfillment	Receive order for shipment Return shipment status	Extended enterprise
Bank	Receive credit card transaction Receive funds transfer requests	Extended enterprise

Finally, Figure 7-4 shows the overall system context overlaid with the Business patterns identified in Table 7-1.

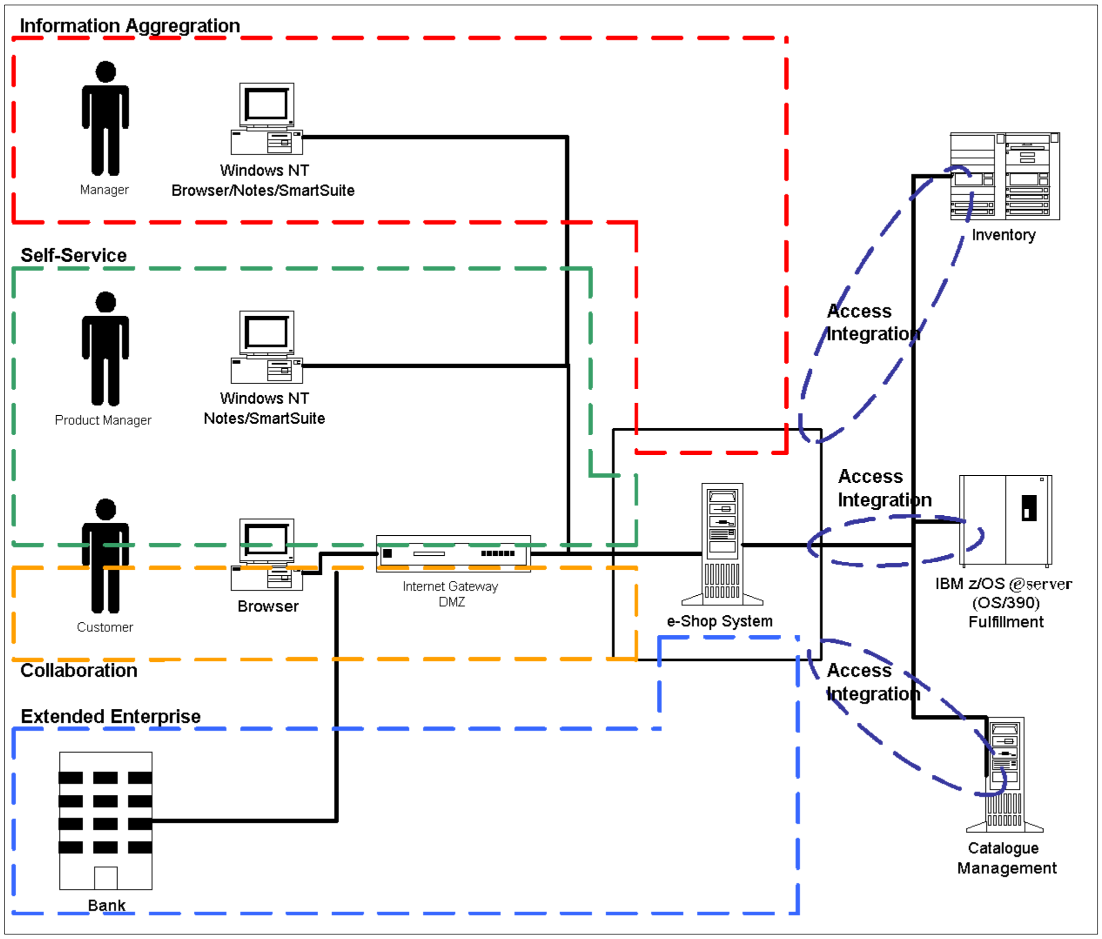


Figure 7-4 System context diagram with Business patterns

7.3 Select an Application pattern

Now that we have identified the Business patterns in this scenario, we need to identify the high-level logical components of the solution and their interactions. This will allow us to choose an appropriate Application pattern.

We begin the analysis by outlining the steps of each new process that the system must support.

7.3.1 Process analysis

Customer registration

As is common in online stores, anyone will be able to browse the e-shop's catalog and even add items to a shopping cart. However, to actually place an order at the site, a user will need to register as a customer.

Figure 7-5 shows the steps involved in registering as a customer.

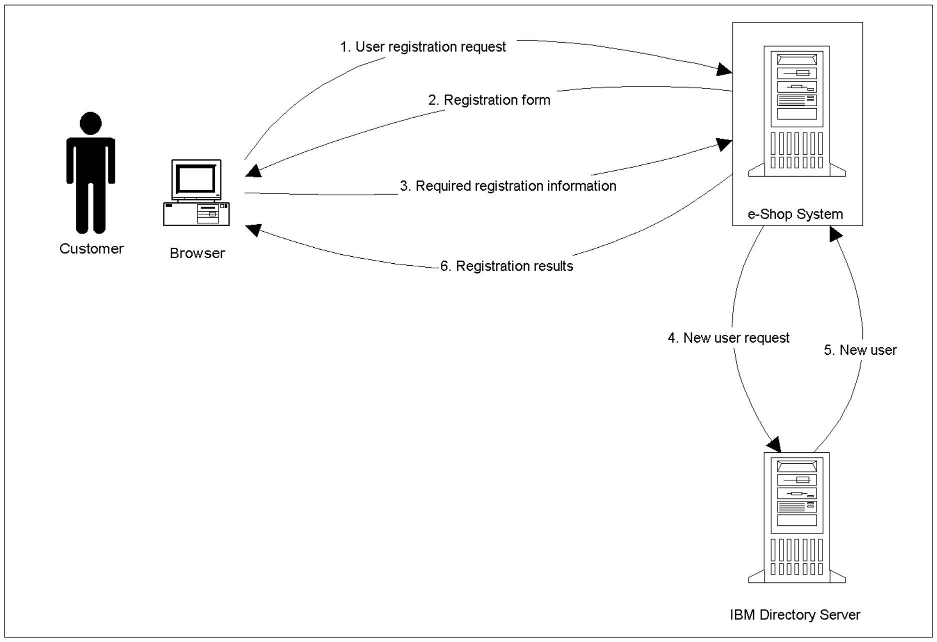


Figure 7-5 Customer registration process

The process proceeds as follows:

1. A user needs to register with the e-shop system before he can place an order. He may do this at any time by explicitly selecting the registration link. In addition, the registration process will begin automatically when an unregistered user attempts to place an order.
2. The e-shop system displays a form on which the user must supply information about himself. This includes some information which is required for order processing, such as name, address, e-mail address, and phone number.
3. After filling in the forms, the user submits the registration to the e-shop system.
4. The e-shop system formats and sends a request to add the user to the IBM Directory Server.
5. The IBM Directory Server adds the user and returns status to the e-shop system.
6. The e-shop system returns registration results to the user. If the registration was initiated automatically as a prerequisite for another operation, that operation is resumed.

Shopping and purchase

The primary purpose for the e-shop site is to sell products to customers. The shopping and purchase process is shown in Figure 7-6.

The shopping and purchase process is discussed in more detail in 7.6.1, "Design guidelines" on page 130.

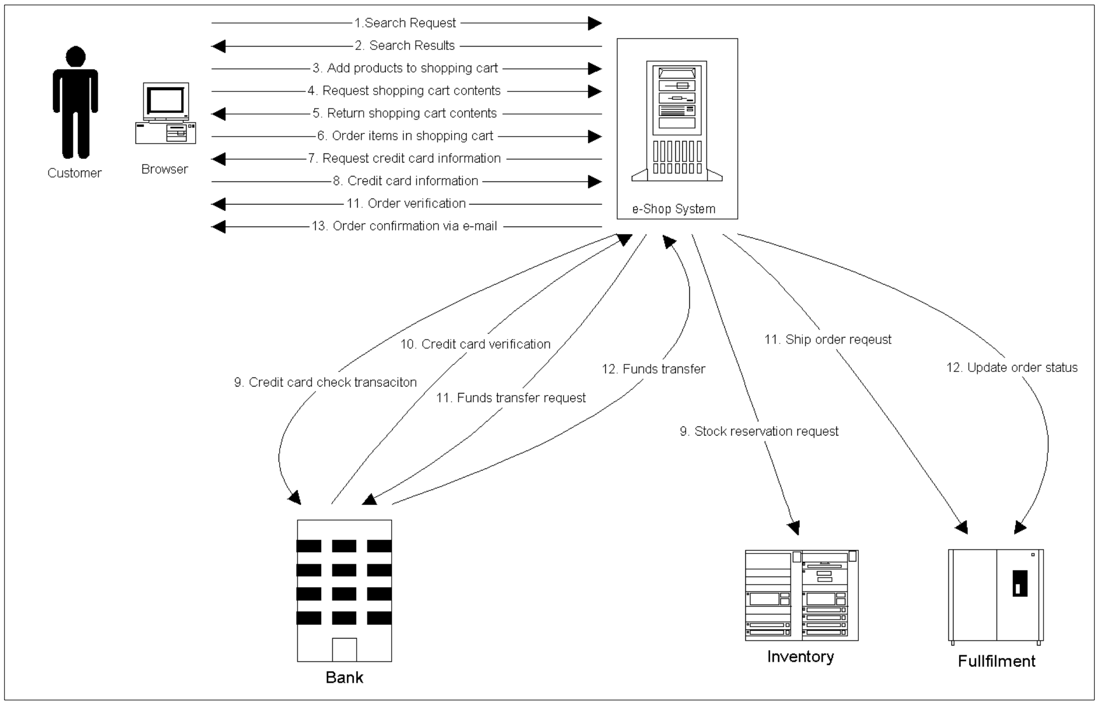

1.Search Request
2. Search Results
3. Add products to shopping cart
4. Request shopping cart contents
5. Return shopping cart contents
6. Order items in shopping cart
7. Request credit card information
8. Credit card information
11. Order verification
13. Order confirmation via e-mail
Customer
Browser
e-Shop System
10. Credit card verification
9. Credit card check transaciton
11. Funds transfer request
12. Funds transfer
9. Stock reservation request
11. Ship order reqeust
12. Update order status
Bank
Inventory
Fullfilment

Figure 7-6 Shopping and purchase process

The following steps describe the shopping and purchase process:

1. The user requests a search for products within the catalog based on some search criteria.
2. The e-shop system returns the search results.
3. The user adds products from the result to his shopping cart. The user may cycle through these steps until his shopping cart contains all of the items he wants to purchase.
4. The user asks to see his shopping cart.
5. The system returns the shopping cart contents.
6. The user indicates that he wants to order the items in his shopping cart.
7. The e-shop system verifies that the user is a registered customer and that the user is logged into the system. (If not, the user is taken through the appropriate steps to register or log in, and then the order process continues.) The system sends a request to the customer for credit card information or verification (depending on whether or not the customer has supplied a credit card as part of his registration information).
8. The user either supplies or verifies the credit card information.
9. The system sends a credit card check transaction to the bank and a stock reservation request to the inventory system.
10. The bank responds to the credit card check.
11. If the credit card check fails, the e-shop system tells the inventory system to remove the stock reservation. Otherwise, the system sends a ship order request to the fulfillment system and a funds transfer request to the bank. The user's browser is sent to an order verification page.
12. The bank transfers funds and the system updates the order’s transaction records.
13. The system sends an order confirmation e-mail to the customer.

7.3.2 Order tracking

The order tracking process is shown in Figure 7-7.

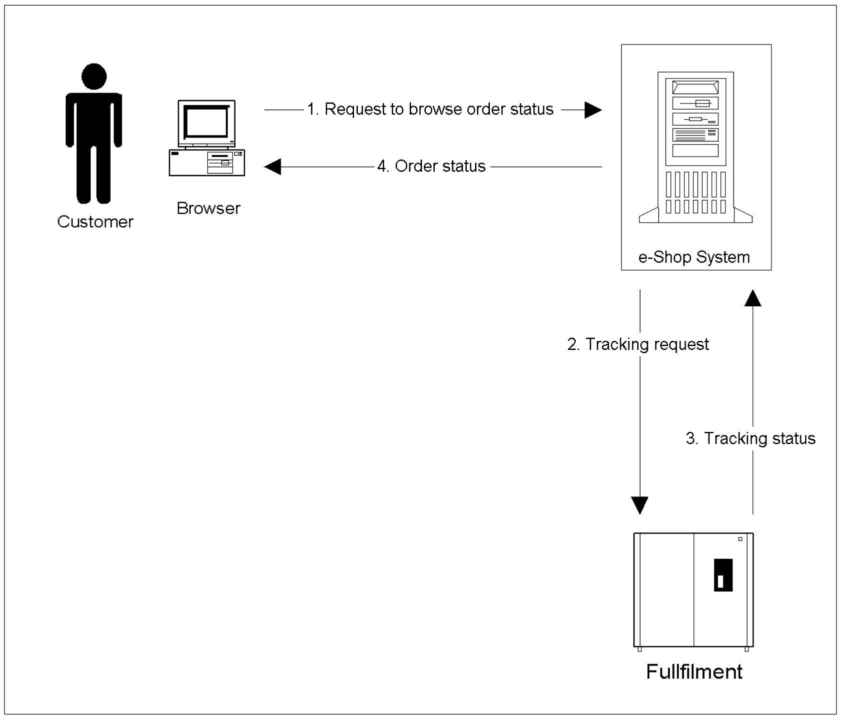


Figure 7-7 Order status tracking process

The steps in order status tracking are the following:

1. The customer logs into the site and requests the status of an order.
2. The e-shop system sends an order tracking request to the fulfillment system.
3. The fulfillment system returns the order tracking information.
4. The e-shop system returns the requested order information to the customer.

Catalog publishing

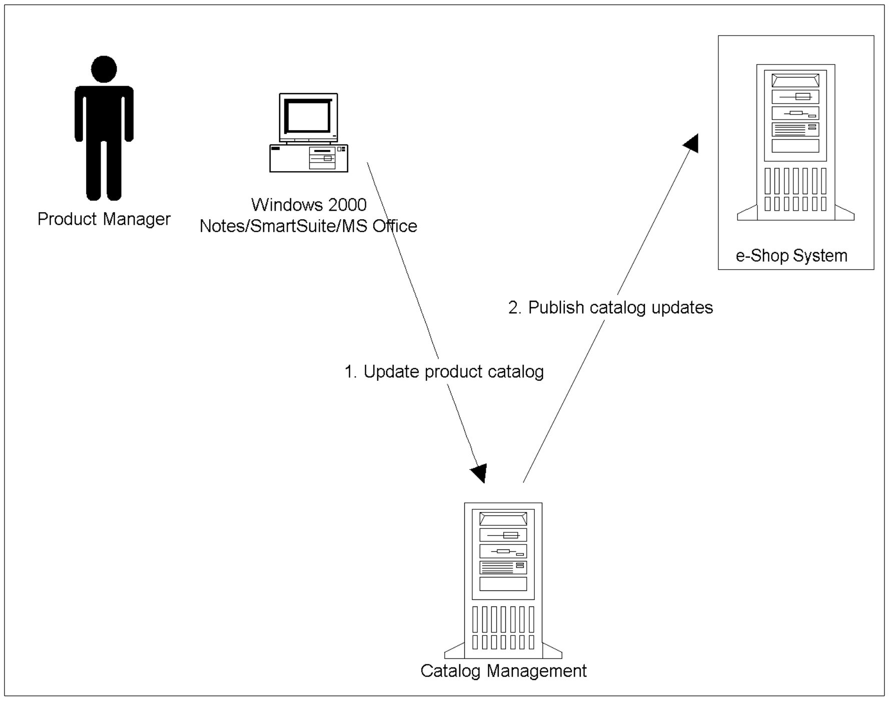

Product Manager
Windows 2000
Notes/SmartSuite/MS Office
e-Shop System
2. Publish catalog updates
1. Update product catalog
Catalog Management

Figure 7-8 Catalog publishing process

The catalog publishing process is shown in Figure 7-8 and described as follows:

1. A product manager updates the products catalog in the existing Domino.Doc product management system.
2. The latest changes in the product management system are automatically published to the e-shop's product catalog.

Reporting

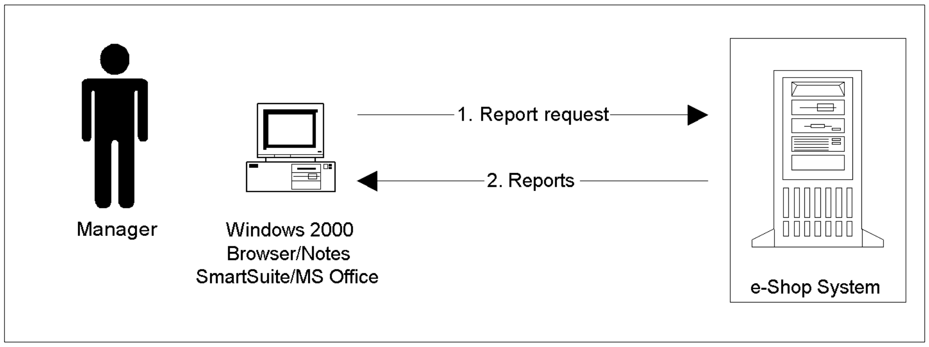


Figure 7-9 Reporting process

The reporting process is shown in Figure 7-9 and described as follows:

1. A manager submits a request for a business report.
2. The e-shop system generates the requested business report and returns it to the manager.

7.3.3 Enterprise-out application pattern

Based on our analysis, the appropriate Application pattern for the e-shop application is the *Enterprise-out* pattern. The pattern is shown in Figure 7-10.

The Enterprise-out pattern is one of the Application patterns that can be applied to the electronic commerce Business pattern. It is discussed in the redbook *e-commerce Patterns for Building B2C Web Sites Using IBM WebSphere Commerce Suite V5.1*, SG24-6180, and on the IBM developerWorks pattern site at:

`http://www.ibm.com/developerworks/patterns`

The following discussion highlights the aspects of this pattern that are relevant to the e-shop application. Visit the Web site for the complete pattern description.

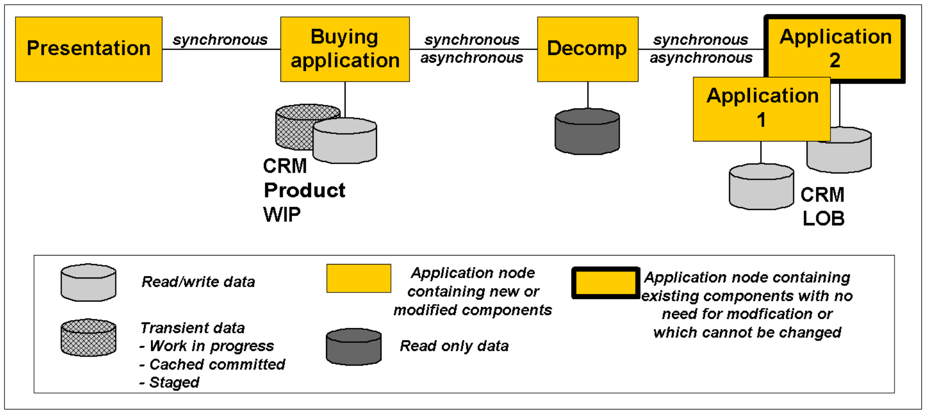


Figure 7-10 Enterprise-out application pattern

Introduction

The Enterprise-out application pattern applies to scenarios that require integration with legacy or third-party applications. The new application is not built as a stand-alone solution; it needs to integrate with existing applications.

This clearly fits the situation with the e-shop application where inventory, fulfillment, and catalog systems that already exist need to be incorporated into the new application.

Business driver

The enterprise-out pattern uses Web technology to sell goods directly to the consumer. Wherever possible, existing order management and fulfillment systems are reused. As organizations become more sophisticated in their implementation, they typically personalize the consumer's experience.

The business goal of integrating the new application with the existing inventory and fulfillment systems is directly aligned with the first business driver. As mentioned before, personalization is not a goal for the first implementation.

Key features

The Enterprise-out application pattern integrates an online buying application with existing enterprise systems. The pattern also addresses maintenance of data on that tier. Such data includes:

- Product and catalog data supporting the shopping process

- Customer-related data for personalization and registration (often duplicated in the enterprise systems on the third tier)
- Work-in-process data, such as data supporting shopping cart functionality, prior to submitting the orders for processing

In addition, the pattern provides the ability to use a form of middleware to access enterprise systems on the third tier. The middleware routes the messages to back-end systems. The middleware might decompose a request into several back-end interactions, and might serve as a broker, potentially communicating with the systems of other companies.

Finally, the pattern supports access to back-end fulfillment systems that perform functions such as inventory, order management, pricing, shipping, tax calculation, and credit processing.

The interactions between the shopper and the middle-tier application are synchronous during the buying process, but asynchronous confirmation messages are often sent back to the user, usually by e-mail.

All of these features, except for personalization, directly address the business goals for the e-shop application.

Considerations

- Consider how this pattern will be deployed to avoid systems management complexity. Complexity arises when frequently updated corporate data resides on more than one tier, with the second and third tiers both within the same organization but physically distributed. For example, synchronization of backups may be cumbersome.

  Except for the online product catalog data, all data for the e-shop application is stored in the same tier. The online product catalog is deployed in a different tier, but the data in the catalog is read-only.

  There may be some concerns with the coordinated backup of the information from the fulfillment and inventory systems, but this is not affected by the additional e-shop application components.

- If different IT organizations are developing the new application and maintaining or changing the existing systems, the development might be difficult to coordinate. Development can be especially difficult if the interfaces between the new and the existing systems are not properly defined and documented.

  Laura’s Gadgets has only one small IT organization, so coordinating the interfaces should be relatively easy.

- The new application might require changes to existing production systems. This is always a critical task, especially when the back-end systems or third-party applications are mission critical. Building a test system for the existing applications that need to be changed is a good way to avoid production system failure while developing and testing the new e-business application.

  The proper product choices should keep required changes to the existing system to a minimum. The current interfaces to the bank, inventory, and fulfillment systems should be usable with minimum changes. Some new development will be required to propagate product catalog changes from the Domino.Doc system to the online product catalog.

  A test environment will be provided to ensure that the production systems are not affected during the development process.

- Obtaining skilled resources that can change the third tier application is often a problem. Many times, these are old applications, and finding someone who understands them well enough to change them might be difficult.

We are not expecting that any changes will be necessary to the legacy applications, other than those required to populate the online product catalog from Domino.Doc.

- The creation and management of the data on the second tier, and any required synchronization with the back-end systems, is often a major effort.
- The security of any customer data residing on the second tier is also a critical factor.

Security issues are discuss in more detail in 7.7, “Security considerations” on page 135

7.4 Review Runtime patterns

The Enterprise-out application pattern has one basic Runtime pattern and three variations.

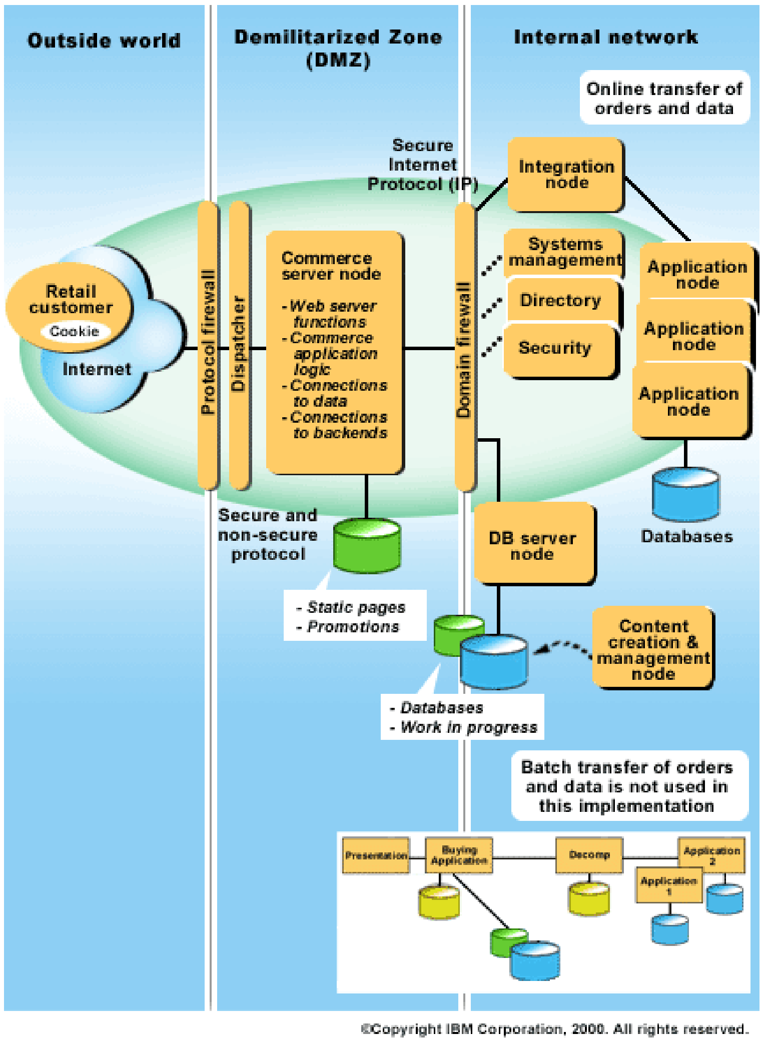


Figure 7-11 Enterprise-out application pattern: Runtime pattern

The enterprise-out basic online shopping process, as shown in Figure 7-11, supports the following processes:

1. From a Web browser client, the shopper connects to the commerce site by entering the Web site URL and does one of the following:
   a. Logs into the commerce site if the user is recognized as a registered shopper from a profile on the database server
   b. Browses the site anonymously
   c. Enters profile information to become a registered user
2. The user then interacts with the pages of the site. These are either static pages from the commerce server or pages dynamically built with information from the database server.
3. The user adds items to a shopping cart. The data for the shopping cart is stored on the database server along with required session state information. A cookie is sent to the client browser. This cookie helps the commerce server track the progress of the customer interactions on the commerce site and connect users with their shopping cart.
4. When the shopper wants to buy (or check out the shopping cart) one of two implementations is taken:
   a. An interaction is performed through the integration node to access the back-end application nodes (such as order processing, pricing, inventory, credit, and shipping) and immediately provide feedback to the shopper such as confirmations and delivery dates. This is the preferred approach.
   b. The less preferred approach is to store the submitted order on the database server for later submission using batch processing. Acknowledgment that the order has been submitted is sent back to the shopper from the commerce server to the Web browser. An e-mail confirmation of delivery, out of stock conditions, and credit problems is later sent by the back-end processing systems. From a user's viewpoint this approach lacks features such as immediate inventory validation, order completion, and estimated shipping details. From the company's viewpoint, this approach provides less direct control of order completion and shipping processes. This approach is similar to the Web-up approach, except that it provides more automated linkage to the batch processes.

Typically, the logon process uses registration data on the database node to identify registered shoppers and encryption technology, such as Secure Socket Layers (SSL), to protect sensitive transmissions (for example, credit card or address information).

The enterprise-out Runtime pattern variation 1, shown in Figure 7-12, is similar to the base pattern except for two modifications:

1. The application server node is moved to the Internal network.
2. A Web server, which can serve static content as well as redirecting requests to the application server node, is added to the DMZ.

Enterprise-out Runtime pattern variation 2 adds load balancing, while variation 3 provides for outsourcing the commerce site, immediate feedback to customers about inventory availability and order status, and support for call center integration. Since satisfying the business goals for the e-shop application does not require any features from variations 2 and 3, these variations are not discussed further here. For more details on these two variations, see the Developerworks Patterns Web site.

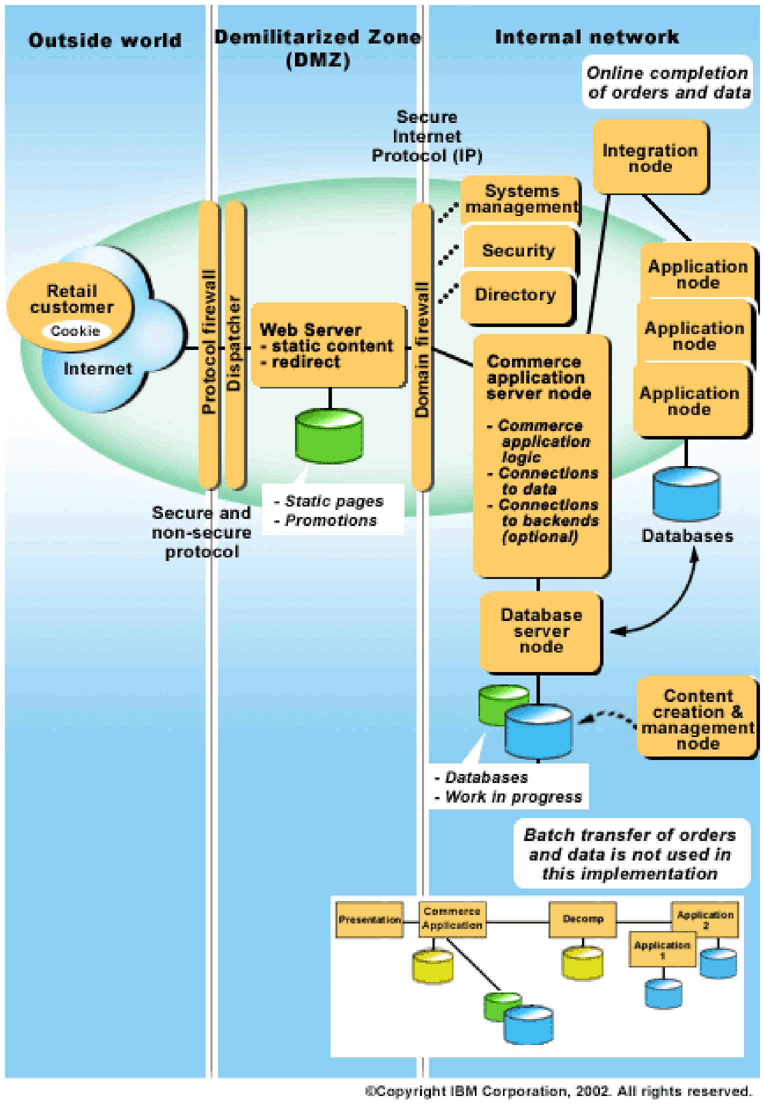


Figure 7-12 Enterprise-out application pattern: Runtime pattern, variation 1

For the e-shop application, we are going to use the basic enterprise-out Runtime pattern, but modify it to include a Web redirector similar to that provided for in variation 1. We chose this pattern because the process it supports, as outlined previously, is virtually identical to the e-shop's shopping and purchase process. The additional Domain firewall provides more security for the existing back-end system, the e-shop's transactional data, and the directory information. The resulting Runtime pattern is shown in Figure 7-13 on page 126. In this diagram, the commerce application server node is represented by a WebSphere Application Server and a Domino Server, the two application servers used in the e-shop solution.

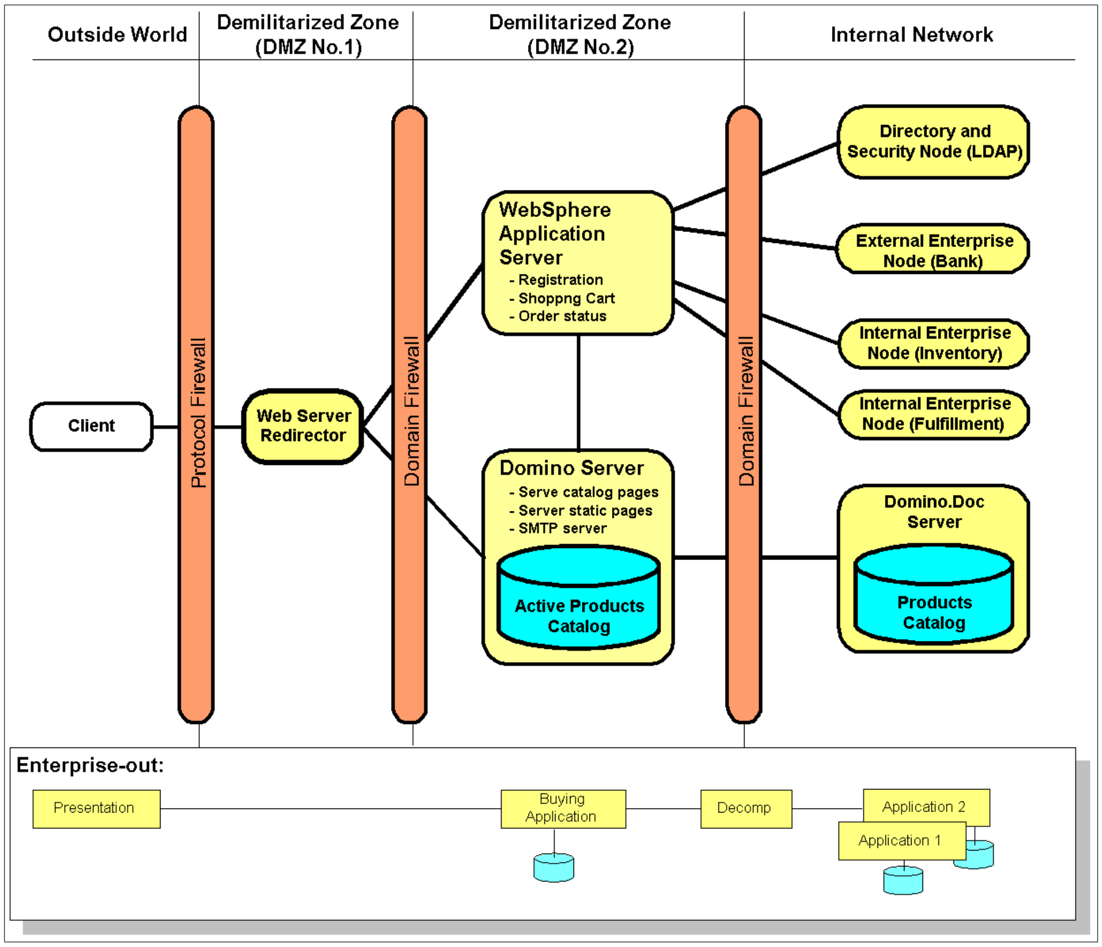


Figure 7-13 Basic Enterprise-out Runtime pattern modified for e-shop application

Although we have added the optional additional firewall between the application servers and the internal network, this pattern is an instance of the hybrid pattern described in 3.2.3, “Web redirector with Domino and WebSphere Application Server” on page 41.

7.5 Review product mappings

The following broad categories cover the processes required to meet the e-shop application goals outlined in 7.2.1, “Business goals for the solution” on page 112:

- Stateless and anonymous information retrieval (user searching and browsing the catalog)
- Information management (catalog data creation, editing and publication)
- Transactional order processing for authenticated customers, including customer data capture and management (customer registration, shopping cart, order processing)

This characterization suggests a mapping of the functionality to products, based on product capabilities, as follows:

- Lotus Domino will provide the online catalog presentation and search facilities.
- In the current IT environment, Lotus Domino.Doc already supplies the catalog information management functionality. Additional publishing functionality will be added as part of the e-shop application.

- WebSphere Application Server will provide all the transactional aspects of the solution, including shopping cart, order tracking, customer registration, and application integration.

In addition to these products, which directly address the business goals, the following products will be used to provide underlying functionality and to integrate the various new and existing products:

- IBM DB2 Universal Database
- IBM HTTP Server
- IBM Directory Server
- IBM MQ Series

Figure 7-14 shows all the product mappings and protocols applied to the e-shop Runtime pattern. A discussion of the role of each of the products follows.

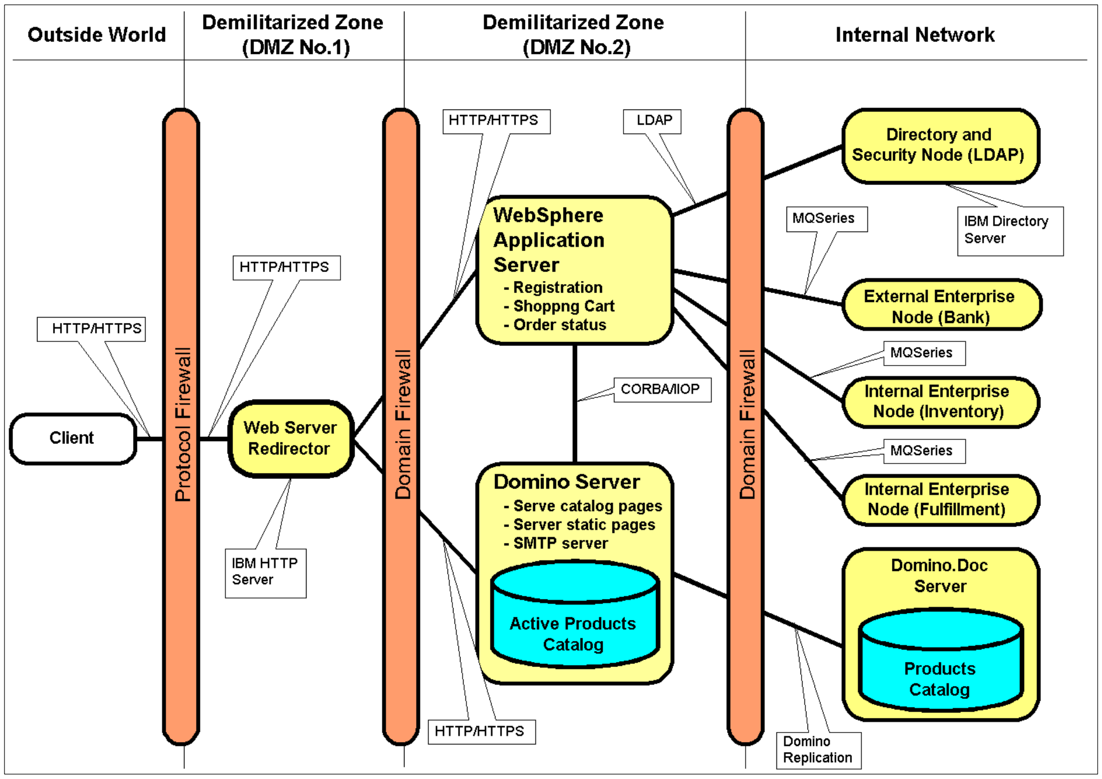


Figure 7-14 Product and protocol mappings

7.5.1 Lotus Notes Domino

In the e-shop application, Domino provides the static pages of the Domino-based Web site, including access to the active product catalog. Domino also serves as the platform for the Domino.Doc system. The Domino.Doc system and the Domino server it runs on are already a part of the IT environment. A separate Domino server is added to support the static Web site pages and the active product catalog. Having two servers simplifies the management of the very different security requirements of these two functional areas. The Domino Web server resides in the second demilitarized zone while the Domino.Doc server resides in the internal network environment

7.5.2 Domino.Doc

Domino.Doc provides all of the content management required for the catalog. Product managers create the catalog content from their Notes client or import content from tools such as Microsoft Word or Adobe Acrobat using the Open Document Management Architecture (ODMA). Once created, the content goes through a predefined approval process before being published. The publishing process is handled by agents that add catalog items from the products catalog in Domino.Doc to the active catalog running on the Web-serving Domino server. Products are added to and removed from the active product catalog based on starting and ending dates and times, which are stored as part of each product document.

7.5.3 IBM WebSphere Application Server

The IBM WebSphere Application Server is installed in the second DMZ and is responsible for the following functionality:

- User registration and login
- Shopping cart processing
- Coordinating the process of ordering, including the use of MQ Series to access the existing internal and external enterprise systems as necessary

As part of the ordering processing, the WebSphere server records a transaction log for all purchases made through the site. This log is maintained as part of the two-phase commit process between WebSphere and MQ Series connected systems.

The interaction with MQ Series is described in detail in the Redbooks *Self-Service Applications using IBM WebSphere V4.0 and IBM MQSeries® Integrator,* SG24-6160 and *Business-to-Business Integration Using MQSeries and MQSI, Patterns for e-business Series,* SG24-6010.

7.5.4 IBM DB2 Universal Database

The IBM DB2 Universal Database is already part of the existing IT environment. It serves as the storage medium for the IBM Directory Server. As part of the e-shop application, it is also used to store the purchase transaction records created during the ordering process. The DB2 databases live in the internal network.

7.5.5 IBM HTTP Server

The IBM HTTP Server (IHS) is used within the first DMZ and is configured to serve as a Web redirector to both the Domino and the WebSphere servers. All access to the e-shop Web site is through this server. IHS provides excellent throughput and speed, and is well suited to its role in the environment.

7.5.6 IBM Directory Server

The IBM Directory Server provides an LDAP-compatible directory service that is integrated with WebSphere to provide the customer authentication required for online ordering on the e-shop site.

7.5.7 IBM MQ Series

IBM MQ (Message Queue) Series is used to carry data to and from the online buying site to:

- Inventory: to update the stock-on-hand figures

- Fulfillment: to fill orders and ship them to the customer
- The bank: to process customers' credit card details and provide payment for ordered goods

7.5.8 Platform

The key considerations in deciding what platform to choose are:

- Software availability for each platform
- Scalability of the platform
- Robustness of the platform
- Security of the platform

When we consider these factors, there are a large number of platforms that we have available to us.

Our selected Web application server and collaboration server (IBM WebSphere Application Server and Lotus Domino, respectively) are both available on platforms ranging from Windows 2000 and Linux microcomputers all the way through to z/OS (OS/390) mainframes. Our options are therefore not restricted by software availability for individual platforms.

Exactly the same can be said about scalability, with vertical scalability provided by implementing the system on more powerful hardware. The addition of horizontal scaling tools such as WebSphere Edge Server, Domino clustering, and WebSphere cloning ensures good scalability when demand starts to push the limits of the existing topology. See Chapter 9, "Scalability and redundancy" on page 161 for more information about scalability.

Platform robustness as a general rule increases as the platform selection moves from Microsoft Windows NT through to mini, mid-range, and mainframe computers. The selection of platform on this basis will be dependent on the e-shop application's requirements for system availability.

While each of the servers in the environment could be installed on different platforms, from an administrative perspective we recommend selecting one platform and using that for all of the servers. That way, systems administrators do not need to be multi-skilled in order to manage the various servers.

7.6 Review guidelines and related links

This section does not contain a full review of the design, development, and management guidelines for the Laura's Gadgets e-shop, but rather concentrates on two design considerations specific to the e-shop application and a review of performance guidelines.

The design guidelines presented in this section discuss the catalog publishing process and present a detailed walk-through of the information flow for the shopping and purchase use case outlined in 7.3.1, "Process analysis" on page 117.

See the Patterns Web site for more design, development, and management guidelines that should be considered when implementing a solution based on the Enterprise-out application pattern.

7.6.1 Design guidelines

Publishing the catalog

The product catalog is maintained by the Domino.Doc application. A publication mechanism is provided to support the publication of the product catalog to a staging environment. After a product manager has reviewed and approved the catalog on the staging site, the contents are copied from the staging environment to the production environment using Domino replication.

The staging environment contains a platform that is functionally equivalent to the production platform. This allows new releases of the catalog to be tested and validated before the catalog is released to the production system.

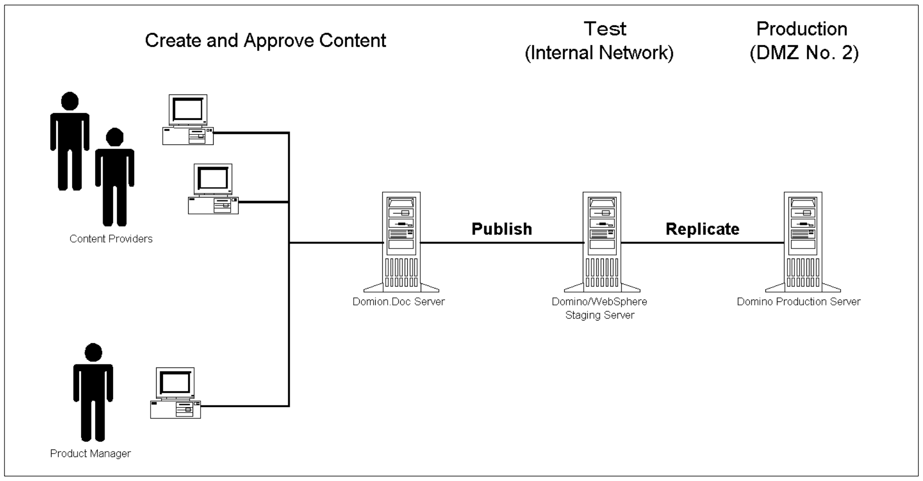


Figure 7-15 Publish, test, and replicate

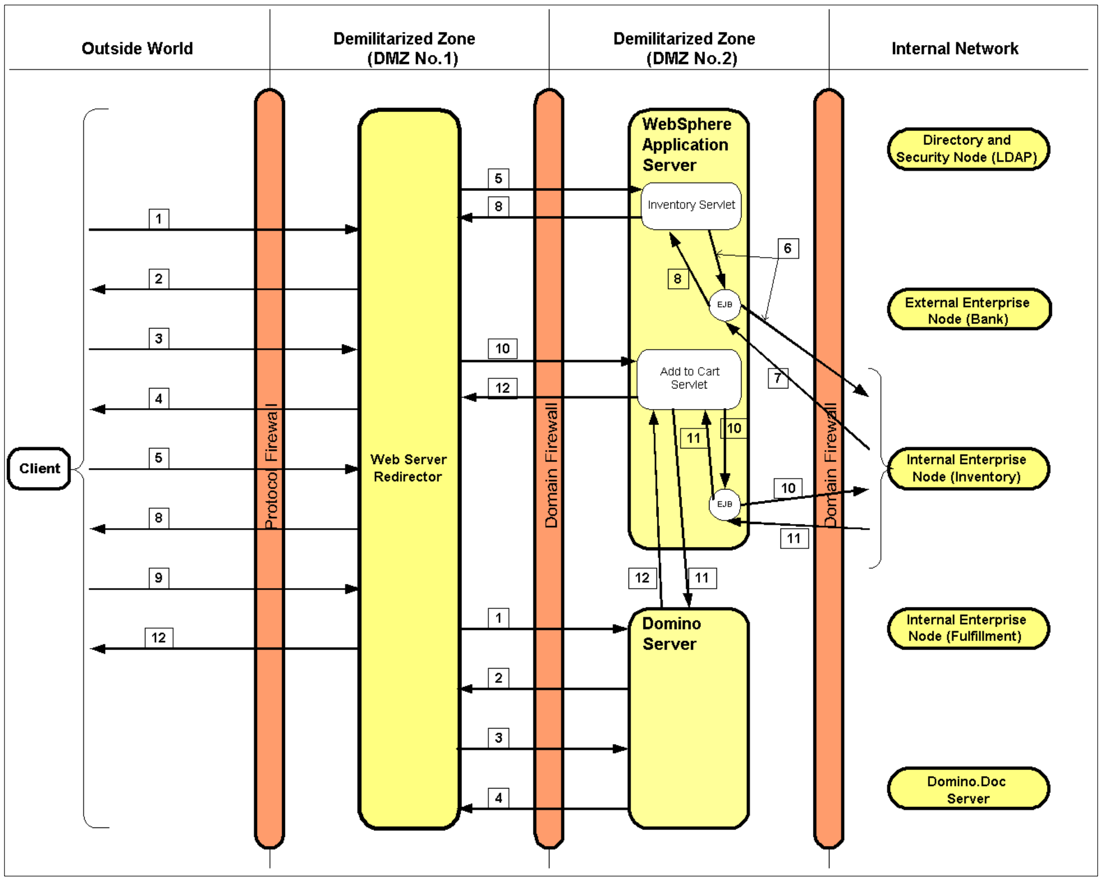
The catalog publication mechanism needs to ensure that product IDs are consistent with the inventory system and are invariant (for a given product) across the publication of new versions of the catalog. This ensures that any in-process orders are not affected by the replication process.

Shopping and purchase information flow

This section provides a walkthrough of the information flow for the shopping and purchase use case to show one way in which the Enterprise-out application pattern can be applied to a process use case. The shopping and purchase use case is shown in the diagram in Figure 7-6 on page 118 and is described in the associated text.

The walkthrough is divided into two parts. The first, shown in Figure 7-16 on page 131, covers the shopping portion of the use case; the second, shown in Figure 7-17 on page 133, covers the purchase process.

information flow for the shopping process



Outside World
Demilitarized Zone (DMZ No.1)
Demilitarized Zone (DMZ No.2)
Internal Network
Client
Protocol Firewall
Web Server Redirector
Domain Firewall
WebSphere Application Server
Inventory Servlet
EJB
Add to Cart Servlet
EJB
Domino Server
Domain Firewall
Directory and Security Node (LDAP)
External Enterprise Node (Bank)
Internal Enterprise Node (Inventory)
Internal Enterprise Node (Fulfillment)
Domino.Doc Server
1
2
3
4
5
6
7
8
9
10
11
12

Figure 7-16 Information flow for the shopping process

1. The customer visits the site by clicking on a link to the site or entering the site's URL into a browser. The request is received by the Web redirector. Based on its configuration information, the Web redirector forwards this request to the Domino server.
2. The Domino server returns the home page for Laura's Gadgets e-shop. In addition to the typical home page information, this page contains links to access various mechanisms for browsing the active catalog. These include:
   a. A search box and button to allow customers to locate products based on names or words in their descriptions
   b. A link to the route of the site's product taxonomy: this provides a navigation mechanism similar to that found on yahoo.com
   c. A button to send the customer to the Weekly Specials page on the site
3. The customer clicks the Weekly Specials button. Based on the URL for the button's link, the Web redirector sends this request to the Domino server.
4. The Domino server returns the Weekly Specials page. For each item on special this week, the page contains the item name, picture, description, price, availability and an Add to Cart button. The availability information is displayed in an iFrame (or iLayer for Netscape). The availability iFrame for each item contains a URL which retrieves the inventory for that item, as detailed in the next step.
5. While displaying the page, the browser processes the availability iFrames. For each iFrame on the page, the browser sends the URL to the redirector. Based on the format of

the URL, the redirector routes each request to the WebSphere server. For each inventory request that WebSphere receives, it invokes the inventory servlet.

6. The inventory servlet creates an inventory EJB for the item. The EJB then uses MQ Series to request the inventory information for the associated item from the inventory system.
7. The inventory system returns the inventory information to each EJB, again using MQ Series.
8. The inventory servlet executing for each item retrieves the inventory for that item from the EJB, and returns it as part of an HTML response that is displayed in the iFrame for that item.
9. The customer locates an item that he wants on the Weekly Specials page. He clicks the item's Add to Cart button. This action sends a POST to the redirector which identifies the item to be added to the cart. The request also includes information about the user's current page.
10. The redirector forwards the request to WebSphere. WebSphere calls the add to cart servlet. The add to cart servlet stores the current page information so it can return the user to that page when the add to cart processing is completed. The servlet then constructs an inventory EJB. This EJB is the same one used to populate the iFrame when displaying the item information described previously. Just as before, the EJB uses MQ Series to get the inventory count for this item from the inventory system.
11. The inventory system returns the inventory count for the item to the EJB. If the item is not available, the servlet redirects the user's browser to an error page that explains that the item is not available. The error page redirects to the Weekly Specials page (or whatever page the customer was on when he clicked the Add to Cart button) when the customer acknowledges the error. Otherwise, the servlet uses CORBA/IIOP to request the object's price from the catalog on the Domino server.
12. The Domino server returns the item's price. The servlet checks the session to see if it contains a shopping cart. This is the first item the customer has selected, so there is no shopping cart. The servlet creates a new shopping cart object, associates it with the session, and adds the information for the selected item to it. Finally, the servlet redirects the customer to the page he was on when he clicked the Add to Cart button.

The customer can now repeat step 9 to add additional items, return to the home page and select another method of navigating the catalog to continue shopping, or check out. The information flow for the checkout, or purchase process, is shown in the next section.

Note that up until this point, there has been no exchange of personal data so there has been no need to encrypt any of the traffic between the browsers and the application servers.

Purchase information flow

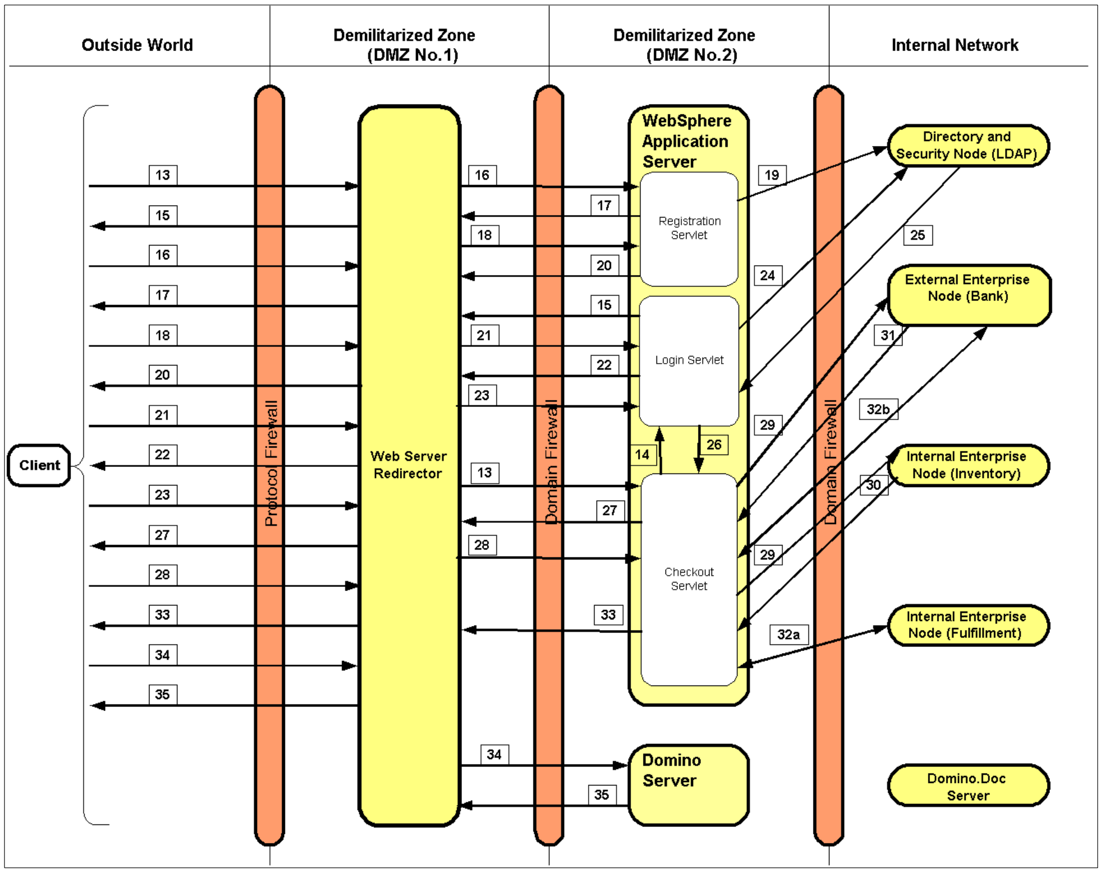

Outside World
Demilitarized Zone (DMZ No.1)
Demilitarized Zone (DMZ No.2)
Internal Network
Client
Protocol Firewall
Web Server Redirector
Domain Firewall
WebSphere Application Server
Registration Servlet
Login Servlet
Checkout Servlet
Domino Server
Directory and Security Node (LDAP)
External Enterprise Node (Bank)
Internal Enterprise Node (Inventory)
Internal Enterprise Node (Fulfillment)
Domino.Doc Server
13
15
16
17
18
20
21
22
23
27
28
33
34
35
14
19
24
25
26
29
30
31
32a
32b

Figure 7-17 Purchase information flow

13.The customer has added all of the items he wants to purchase to his shopping cart according to the process described in the previous section. He is now ready to complete his purchase, so he clicks the Checkout link. Before this request (which includes information about the current browser page) is sent to the Web redirector, the browser is switched to secure mode. Based on the format of the URL, the redirector forwards the request to the WebSphere Application Server. The WebSphere Application Server calls the checkout servlet.

14.The checkout servlet stores the current browser location for later use. It then checks the session for a shopping cart. If there is no cart or if the cart is empty, it redirects the browser to a page that explains that the system cannot checkout an empty shopping cart. In this case, the customer is returned to his previous catalog page when he acknowledges the error.

Having verified that the session contains a non-empty shopping cart, the servlet now checks the session to see if the customer is logged in. In this scenario, he is not, so the checkout servlet calls the login servlet.

15.The login servlet returns a login page to the user. The page contains fields for username and password, as well as buttons to log in and to register.

16.The customer has never registered, so he clicks the register button. This sends a request to the redirector, which forwards it to WebSphere. WebSphere invokes the register servlet.

17.The register servlet returns a registration form to the user. It contains fields for the customer's name, address, e-mail address, desired username and password, and a submit button.

18. The customer fills in the fields on the form and clicks submit. The browser sends a POST request to the redirector, which forwards it to WebSphere. WebSphere invokes the complete registration method of the register servlet.
19. The register servlet stores the customer's information in the LDAP directory.
20. The register servlet redirects the customer's browser to a page which confirms that the registration is complete.
21. The user acknowledges the confirmation. This acknowledgement is routed through the redirector to WebSphere. WebSphere then invokes the login portlet.
22. The login portlet sends the login page to the customer's browser.
23. The customer provides his new username and password and clicks the Login button. This sends a request through the redirector to WebSphere. WebSphere calls the login servlet.
24. The login servlet submits the user information to the LDAP directory for validation.
25. The LDAP directory validates the user, and the login servlet then associates the shopping cart in the session with the validated user.
26. The login servlet remembers that the customer was in the process of checking out and invokes the checkout servlet.
27. The checkout servlet checks the session to see if the customer is logged in. In this case he is, so the servlet sends a page to the customer to collect his credit card and shipping information.
28. The customer supplies his shipping and credit card information and clicks submit. This sends a POST request through the redirector to a method of the checkout servlet in WebSphere.
29. The checkout servlet uses MQ Series to send an approval request for the customer's credit card to the Extended Enterprise bank system and a stock reservation request to the inventory system.
30. The inventory system responds to the stock reservation request.
31. The bank responds to the approval request. If the request fails, the servlet:
    a. Tells the inventory system to cancel the stock reservation
    b. Routes the customer though an error page
    c. Redirects the user to the credit card information page to give the customer a chance to correct any errors he may have made while specifying this information

    We assume here that the request is approved, so the checkout servlet stores the returned reference number.
32. The checkout servlet performs these actions within the boundaries of a two-phase commit transaction (this could be done using a session EJB):
    a. Create an order and use MQ Series to send an order request to the fulfillment system.
    b. Use MQ Series to send a payment request to the bank based on the credit approval reference number returned previously from the bank.
    c. Wait for the responses from the fulfillment system and bank.
33. The checkout servlet then sends an order confirmation e-mail and redirects the customer's browser to a page which thanks the customer for his order and provides him with his order reference number.
34. The customer acknowledges the order confirmation. This request is redirected to the Domino server.
35. The Domino server sends the site's home page to the customer's browser.

7.6.2 Performance guidelines

The sales platform has higher performance requirements than the JPA scenario and, in keeping with this requirement, the “shopping cart” and purchase transactions are being maintained via the WebSphere platform.

The runtime topology selected will support future transaction growth by using platform replication to provide horizontal scalability among the catalog and application server components, and vertical scalability by having multiple application server instances running on single platforms. The way to achieve this is discussed in Chapter 9, “Scalability and redundancy” on page 161.

The catalog part of the site is accessed anonymously, which means that it can be replicated without the need for maintaining sessions across machines.

7.7 Security considerations

In order to proactively deal with the present and emerging threats and vulnerabilities to business assets over the Internet and intranets, security needs a very special focus right from the beginning of any design process. If this focus is not taken, it is possible that you might need to do a complete redesign to retrofit the necessary security measures. In this section we discuss how the different security measures are applied and utilized in the e-shop scenario. For more information, refer to the Redbooks *Enterprise Security Architecture using IBM Tivoli Security Solutions,* SG24-6014 and *IBM WebSphere V5.0 Security WebSphere Handbook series*, SG24-6573.

7.7.1 Basic security requirements

The security needs of any business are traditionally based on risk assessment and management. How risks are managed is a business decision based on the business’s assessment of the risks involved in not providing various security measures as compared to the benefit achieved by those measures. The factors that influence the decision to choose a particular security architecture depend mainly on the type of application the business is building and the business value of the transactions and data that the application will support. The security requirements for a business generally include:

- Access control
- Flow control
- Audit control
- Credential management
- Integrity

To learn more about a security methodology based on these categories, see “End-to-End Security” in *IBM Systems Journal, Volume 40, No. 3, 2001,* available at:

```
http://www.research.ibm.com/journal/sj40-3.html
```

In the following sections we describe how the topology we chose for this solution addresses the various business security needs. To do so, we start directly with the runtime topology and product mapping.

7.7.2 Security analysis

When operational, this site will be open for all users over the Internet, so this scenario demands the maximum security possible to ensure that the organization's internal resources and data are not compromised. In addition, the site will have to provide a reliable, secure channel and process for acquiring customers' credit card information, personal data, and preferences. The solution also requires a secure gateway for business partners, retailers, and others over the Web. The application architecture should be able to accommodate all of these needs.

Since the site is open to all users, we have to be sure that the data served by the WebSphere and Domino Application servers is well protected. To minimize any possibility of unauthorized access to our network, we have used the double DMZ model in architecting this solution. This double DMZ model helps:

1. Isolate the Web server from the Web application servers
2. Isolate the Web application servers from the internal network
3. Prevent both application servers from being directly accessed over the Internet
4. Hide the organization's internal network and resources from users of the Internet

Figure 7-18 is a copy of the Runtime pattern we developed in 7.4, "Review Runtime patterns" on page 123. Referring to this figure, we see the three firewalls that create the two DMZs. We have the Web redirector within the first DMZ. This is to ensure that any Internet user accessing the system cannot access any other resources beyond this firewall. For any end user accessing via the Internet, it would look like the information is all from the Web server in the first DMZ.

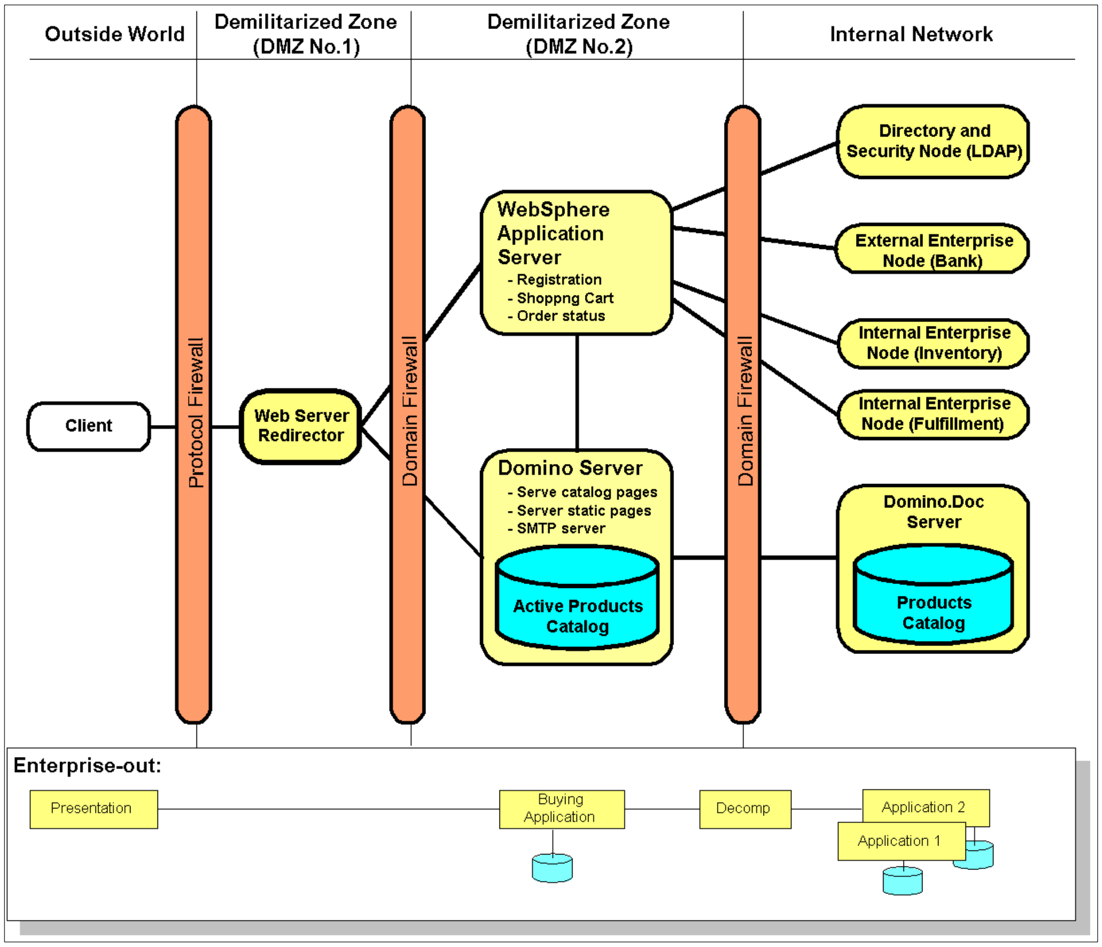


Figure 7-18 e-Shop application Runtime pattern

The Web redirector in the first DMZ receives the Web client requests and redirects them to the appropriate Web application server. The first Domain firewall, separating the first and second DMZs, is configured to only allow requests for the application servers from the Web redirector. Any other traffic is blocked, thus preventing any unauthorized user entry.

The second DMZ hosts the Web and application servers. The optional second Domain firewall provides an extra degree of security between the run-time production environment open to the public and the internal network. Also, the second DMZ helps separate and secure the enterprise data sources from external accesses. The data sources are configured to be accessed only from the Web application server with a preset ID, and are not available otherwise. The firewall separating the Web application servers and the internal network allows only the traffic originating from and destined for Web application servers.

7.7.3 Security requirements for this scenario

In this section we discuss how our topology addresses the various security requirements.

Access control

Laura's Gadgets e-shop is a typical e-commerce site, which partially opens up resources to all Internet users for viewing, but still demands a very high level of security and protection for the entire site and resources. Since the site also stores credit card information for the customers, it has to meet guidelines for protecting and securing that customer data. These guidelines must meet or exceed what is imposed by the authorities in the areas where the e-shop operates. In the United States there are federal guidelines that must be adhered to.

All customers' IDs, passwords, and address information are stored in a secure LDAP located in the internal network, well shielded by the two DMZs.

Domino protects its assets using *access control lists* (ACLs) with multiple levels of security that include organization, server, database, form, field, and so on. Since Domino renders the catalog, home page, and promotions to all Internet users, Anonymous access is allowed for viewing and searching this information.

When a customer checks out, the system automatically prompts the user to register in the LDAP (if they have not already done so). Once registered, the LDAP is used for validating the credentials and authenticating the customer for all further transactions. This includes transactions in and access to WebSphere, Domino, DB2, and MQ series. All authentication information is verified among the application servers and the LDAP over SSL.

WebSphere is configured to use its Security Policy for Servlets and EJBs. Domino and WebSphere could be configured to use session sharing and single sign-on for sharing the authentication information; however, this is not strictly required in this scenario since all the Domino data in the application (the catalog) is public and therefore user authentication is not required.

Access to the Domino.Doc-based catalog and the associated publishing cycle is controlled and protected using Domino ACL policies.

To ensure that a price is not altered by the Internet user and resubmitted, a separate secure request is done by WebSphere to Domino using CORBA/IIOP to retrieve the price of the product the customer requested.

Inventory, fulfillment, and order processing data is stored on secure DB2 servers that restrict access to WebSphere EJBs through MQ and authoring sources. MQ restricts access to its resources at Queue Manager and Queue levels.

Flow control

The Protocol and Domain firewalls, one on each side of the IBM HTTP server, provide a complete separation of the internal network and its data from the Internet. This prevents the possibility of hacking, or intentional or accidental damage to the business data from the Internet.

The first domain firewall is configured to allow only the traffic forwarded by IHS, serving as a Web redirector, to get to the application server. The second domain firewall restricts all data other than that originating from the application servers themselves from going to the internal network resources. This way, even if there is a security compromise on either of the Web Application servers, the rest of the system remains invisible and protected.

SSL plays a vital role when the customer credit card information is collected or retrieved.

End-to-end high encryption and SSL is used for credit card authorization by the bank. The data flow is encrypted all the way from the browser through the Web redirector, the WebSphere Application Server, the MQ Series queues, and on to the Bank Information processing system. All responses on this path are also SSL encrypted.

Audit control

Auditing is turned on at various levels to ensure that customer data is secure. All Web accesses are logged at the Web redirector level. Domino serves the home page and the catalog; in order to ensure that the data in Domino is not compromised, all successful and failed login and access attempts to Domino resources are logged. The firewalls restrict the traffic that goes through them to ensure that configured sources are the only ones that can get past them.

With multiple firewalls hiding the internal network, WebSphere, LDAP, MQ, DB2, and other internal staging and production servers can concentrate on functional auditing and database integrity auditing.

Credential management

Anonymous users are allowed to view the home page of the retail store and to request inventory information for any product. The only operation that requires the user to log in is the process of ordering the items in a customer's shopping cart. WebSphere Application Server challenges any user who attempts to check out to provide a user ID and password over SSL. The SSL connection is also used to secure information that the user supplies when checking out, like credit card number, e-mail address, delivery address, and so on. If the customer is a new user, WebSphere automatically allows the user to register with the LDAP Directory over SSL.

This scenario ensures that all user, group, and related information is centrally and securely maintained by the LDAP, and that all application servers use it for their authentication needs. MQ and DB2 use their own native authentication mechanism with their own internally managed credentials.

Integrity

End-to-end data integrity is ensured by the use of secure authenticated sessions for sending and retrieving data over the network. Transaction logging and the two-phase commit features in DB2, MQ, and WebSphere EJBs ensure that even if inventory is reserved, the order is not completed until the bank returns an authorization code for payment. It also ensures that items are not shipped until payment is debited from the customer's bank.

For additional information about security infrastructure and security considerations in application design and development, see Appendix , "Related publications" on page 187.

7.8 Summary

In this chapter we began by describing the business requirements for extending an existing catalog system to the Web. After reviewing the business goals and current IT environment, we chose the Electronic Commerce composite pattern as the appropriate Composite business pattern for achieving the business goals. After investigating the business processes that the new application will need to provide, we selected the Enterprise-out application pattern. Further analysis led to the selection of the Web Redirector with Domino and WebSphere Hybrid Runtime pattern as the basic Runtime pattern for our solution. This hybrid pattern is the same as the Enterprise-out basic Runtime pattern, except that it specifically includes both Domino and WebSphere as application servers. The Runtime pattern was then modified slightly by adding a second DMZ to further protect the back-end data and systems.

After the pattern selection was complete, we did a partial review of design, development, and management guidelines for the application. We followed this with a discussion of the application's security considerations.

8

Scenario: HR Staffing, Inc.

This chapter illustrates a more complex scenario, and the use of the Business and Application patterns and selection guidelines to design a satisfactory solution. This scenario intentionally does not cover in depth the Hybrid Runtime patterns that previous chapters and scenarios have. There would be simply too many iterations to cover.

Instead, we describe the steps to apply a Composite pattern incorporating both Domino and WebSphere in the solution. As in previous scenarios, we perform the following steps:

1. Select Business patterns
2. Select application topology
3. Review Runtime patterns
4. Review product mappings
5. Review guidelines and related information

All these are considered in accordance with the patterns for e-Business Web site at:

http://www.ibm.com/developerworks/patterns/

8.1 Overview

This chapter describes a growing medium-sized job placement company, HR Staffing, Inc., which has had success in its traditional bricks and mortar business. The company now wishes to connect its existing systems to the Internet, replacing its current Web site, to provide higher service levels for the job seekers and job providers it services.

HR Staffing's goals are to improve organizational efficiency by reducing the time required to process resumes and applications, and to reduce costs by electronically processing this information.

In this chapter we explore the company's current solutions and design a new solution based on strategic enterprise architectural principles to address future growth.

8.2 High-level business description

HR Staffing maintains current resumes of job seekers and current job postings of job providers. *Candidate Advocates* represent job seekers and *Account Executives* represent job providers. Advocates and account executives maintain close relationships with their respective customers and work together to find potential matches. When a match is made and a candidate is placed, fees are charged to the job providers.

8.2.1 Value proposition

In order to understand HR Staffing's business, we look at the company's value proposition. The value proposition is as follows:

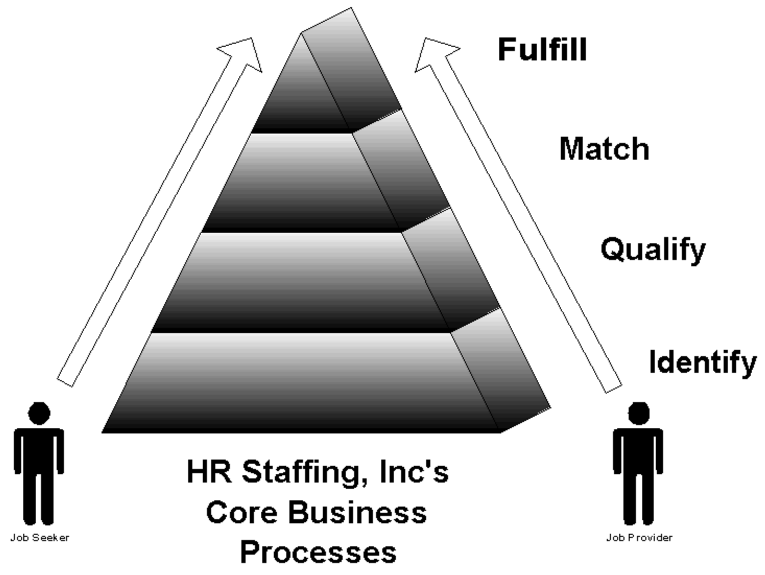


Figure 8-1 HR Staffing's core business processes

- **Identify**: This is the phase of the business model whereby business processes are employed to transform any person or business into an HR Staffing customer. An example is an anonymous user registering as a job seeker and posting a resume. Or, a job provider might register her company and post a job.

- **Qualify**: The phase of the business model where business processes transform identified job seekers and job providers. An example of qualifying job seekers is the service provided by the staffing company to perform employment record verification, background checks, checks of criminal history, education verification, and so forth. For the job provider, qualification might be a credit history check, Dun & Bradstreet verification, and contract term review.
- **Match**: The phase where the business processes align job seekers with job opportunities posted by job providers. This involves the matching of skill sets, interview processes, alignment of the job seeker with environment of the job provider, and negotiation.
- **Fulfill**: The phase where the business process places the job seeker into the new job. This involves pre-employment testing and screenings, billing, and collection.

Through these steps HR Staffing moves job seekers and job providers closer together, while providing value-added services to each along the way. Revenue is generated primarily from services rendered to job providers.

8.2.2 Key business relationships

It is helpful to understand some of the key business relationships that are required for HR Staffing to act upon its value proposition. While other relationships exist, the key business relationships are generically described in Figure 8-2.

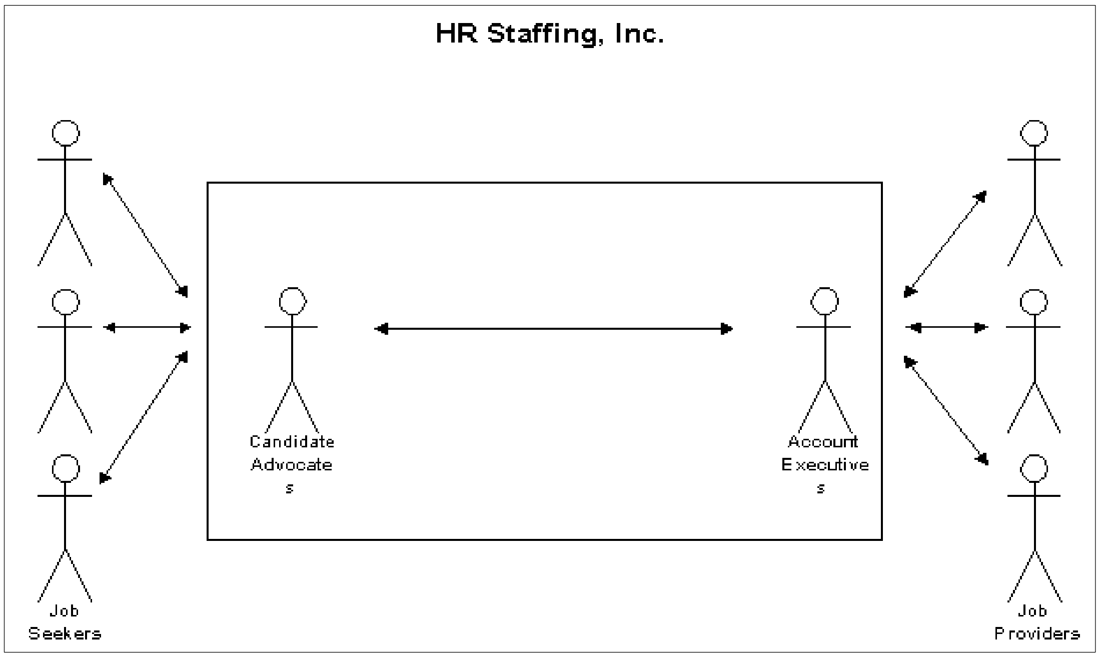


Figure 8-2 Business relationships overview

As the job seekers and job providers move through the business processes in the value proposition, relationships with job seekers are managed by candidate advocates. Likewise, job providers are processed through account executives. HR Staffing has the opportunity to differentiate itself from the competition by providing various service levels for job providers, and this is the primary source of revenues. Job seekers' candidate advocates work with account executives to make placements occur. Speedy completion of this process is key for fulfilling a job order.

8.2.3 Business drivers

The long-term goal is to provide an employment portal for job seekers and job providers. This strategy will enable the following business drivers:

- Grow the number of job seekers and job providers
- Growth in higher margin placements
- Create new services that increase revenue opportunities
- Increase employee productivity
- Decrease cost of maintaining job seeker data
- Improve customer relationships
- Improve data accuracy and speed of collection

8.3 Solution overview

8.3.1 Background

HR Staffing has several office locations throughout North America. Data services have been centralized into a hardened data center to which all offices have T3 WAN connectivity or better. Candidate advocates are office-based, with desktop PCs. Account executives are mobile employees with laptop computers. The company currently has developed several applications for each category of employees using various technologies.

Currently, the company has standardized on DB2 for its Enterprise database platform. There are currently two main applications that employees use: Customer Management System (CustMS) and Candidate Management System (CandMS). Both applications were built using a two-tiered, fat client approach. The systems are business critical. Data accuracy and freshness often determine whether or not a placement is made. HR Staffing wishes to Web-enable significant processes and allow Web-based access to job seekers, job providers, and employees. The company believes that a thin client approach will help reduce long-term costs associated with fat client development and maintenance.

HR Staffing currently runs Lotus Domino as its primary e-mail and groupware technology. Additionally, Domino is used to support its intranet and Internet. The applications deployed for both the intranet and Internet are for mainly informational purposes only, or to assist in business process workflow. HR Staffing has an extensive skill base in Lotus Domino and its related technologies.

HR Staffing is interested in providing these functions on the Internet to its current and potential customers via a browser.

While no specific applications have been designed yet, HR Staffing is also interested in the capability of securely providing business functions over several channels across the Internet/intranet. This includes Web browsers and personal digital assistants for its employees.

While current business processes will grow as more placements are made, the new Web-enabled functions have the potential to show rapid volume growth in a short period of time from the addition of new and existing customers. HR Staffing expects the initial volume of electronic transactions to be small compared to its traditional business processes. Additional, internal training programs will seek to shift employee usage away from fat clients and toward the Web-enabled channels. They also believe that offering this infrastructure to their customer community will provide growth opportunities, and ultimately will become the primary means of transacting business.

8.3.2 Key features

The solution is multi-faceted and must be designed to accommodate core processes at first. As the demand increases, additional services will be provided. For the initial solution to be considered a success, it must provide the following:

- Information for stockholders

  This feature aggregates content from external news content providers regarding the company, approved shareholder content, and stock prices obtained from Nasdaq.

- An electronic guide to help find an office

  This feature is a directory of the company's locations from internal approved content.

- A personalized job seeker/provider experience

  The site will have role-based awareness of the user. Market offerings can be customized to the user based on these roles.

- Electronic job seeker resume posting

  Resume posting will consolidate all of the job seekers' resumes into a standard format. The Candidate Management System will be updated by the job seeker and his or her candidate advocate. This requires the user to complete the registration process.

- Electronic job provider job posting

  Job posting will consolidate all of the job providers' job postings into a standard format. The Customer Management System will be updated by the job provider and his or her account executive. This requires the user to complete the registration process.

- Electronic searching of resumes and job postings

  Job seekers will be able to execute searches upon the job postings database. Job providers will be able to execute searches upon resumes.

- Access to profile management

  Job seekers and job providers will be able to securely manage their respective profiles. Changes will be confirmed by e-mail verification to the request originator.

- Electronic apply for a job process

  Each job posting will be uniquely associated with a job provider. The job posting will include an "apply now" button that will forward a copy of the job seeker's resume anonymously to the job provider, but with known identity to the job provider's account executive.

- Integration with legacy systems for back office processing

  Data entered for job seekers will be stored in CandMS and data entered for job providers will be stored in CustMS.

- Account executive and candidate advocate access to business process workflow information

  Account executives and candidate advocates participate in several verifications for job providers (procedures such as credit verification, delinquent accounts, etc.) and job seekers (criminal history, education history, work history, etc.) respectively. These procedures must be securely Web-enabled.

- Job seeker access to application status

  When the job seeker requests status of his or her application status, the system will respond with an appropriate status for each job that has been applied for.
- Electronic fee payment submission via First National Bank

  The Accounting System will track payments for each job provider's account.
- Instant messaging capability between job seekers and candidate advocates

  During the users' experience, the users can collaborate with one another via chat and e-mail facilities. Chats will be directed toward assisting the user to contact candidate advocates or account executives. E-mails are sent ad hoc and in workflow applications.

8.3.3 Solution overview diagram

We now attempt to translate the text into a pictorial representation of the system.

First, we represent the key business functions:

- Stockholder information
- Find an office
- Search candidates
- Post job
- Manage account
- Search jobs
- Post resume
- Manage profile
- Apply for job
- Business process workflow
- Chat and e-mail

Next we look at pre-defined processes:

- Customer Management System
- Candidate Management System
- Accounting System

Finally, we add any remaining actors.

- News and content providers
- Financial institutions (First National Bank)

The resulting processes and actors are shown in Figure 8-3.

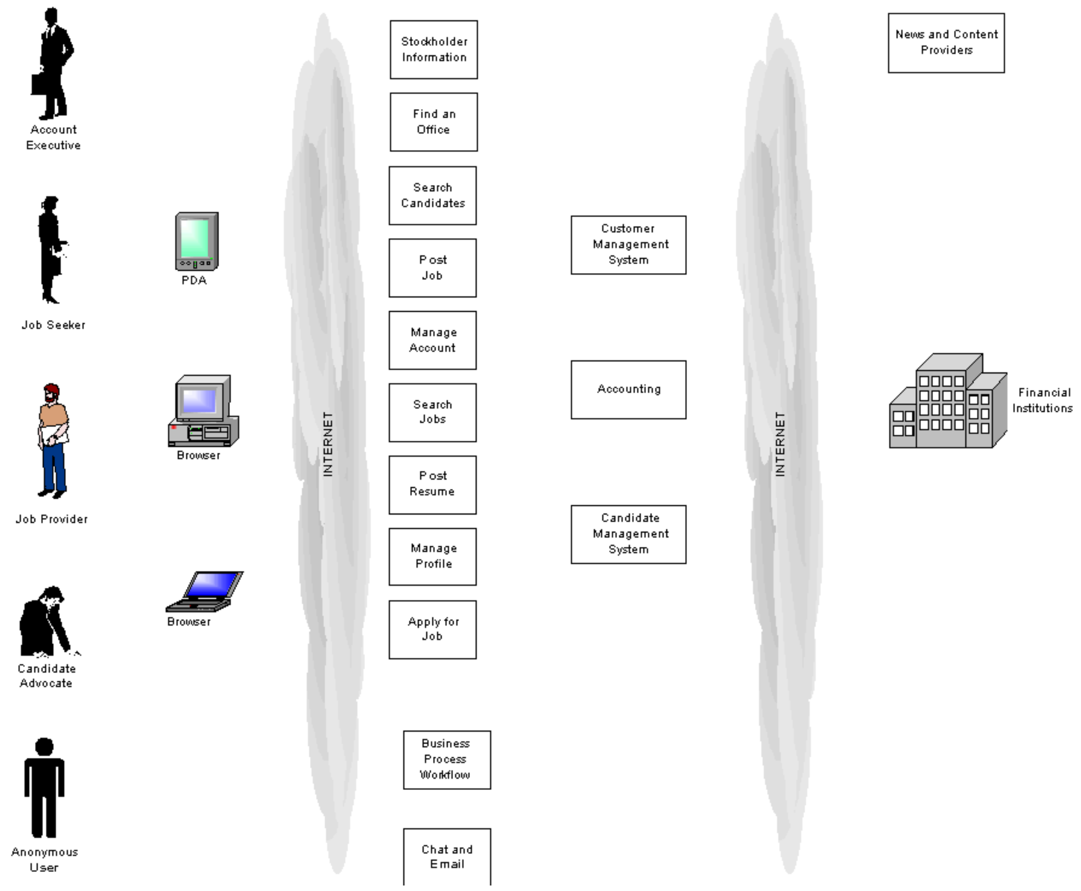


Figure 8-3 Solution overview diagram

The final step in preparing the solution overview is to diagram connections between the individual symbols; the result is shown in Figure 8-4. The business description serves as the basis for this activity.

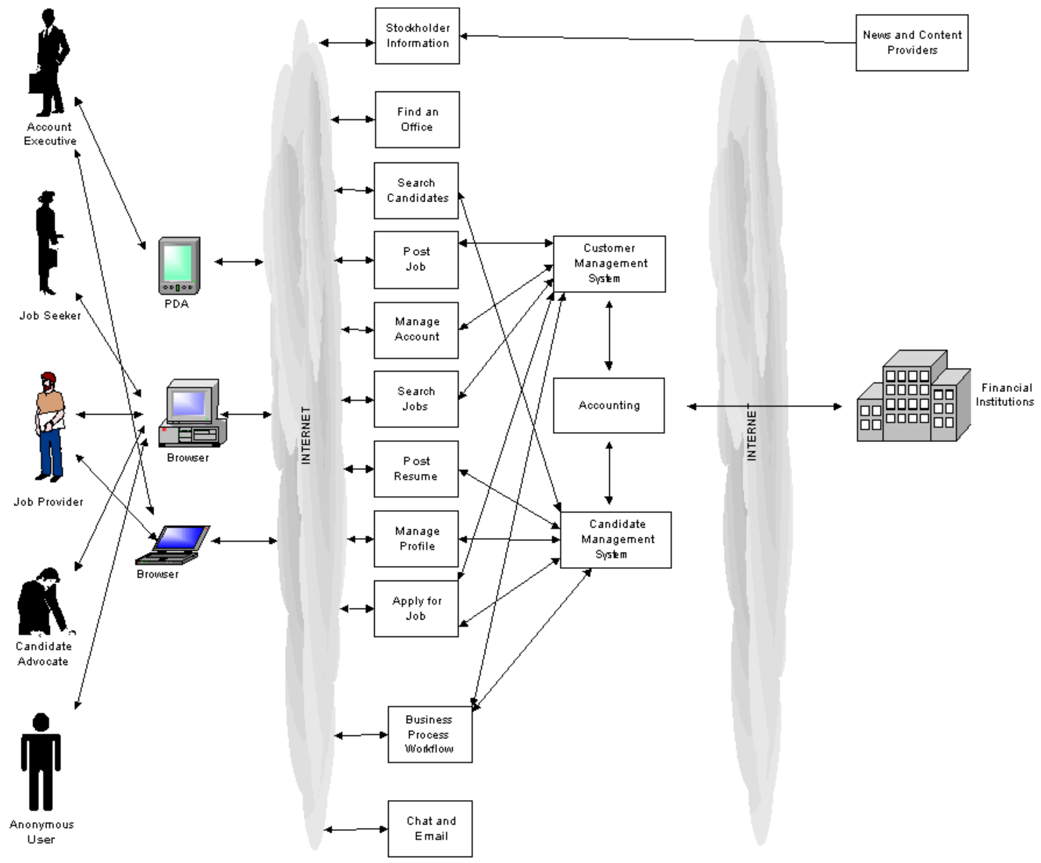


Figure 8-4 Solution overview diagram with business functions, actors, and connectors applied

The solution overview diagram now provides a concise means of communicating the proposed system. It also serves as the key input onto which we apply the Patterns for e-business.

8.4 Identify Business patterns

Review of the Business patterns in comparison to the solution outline diagram in Figure 8-4 revealed a summary of the proposed system. We now map the solution overview to Business patterns.

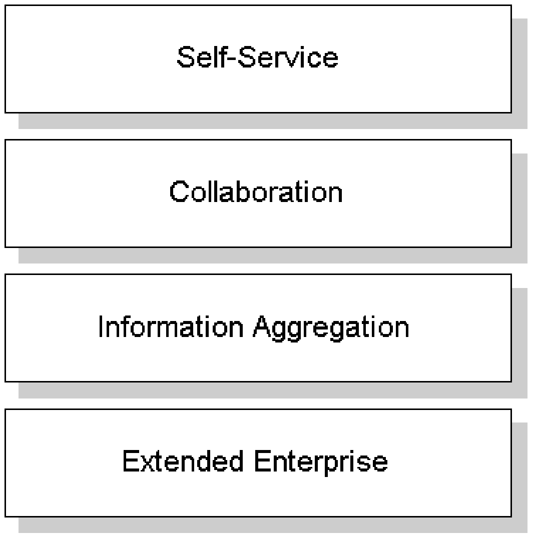


Figure 8-5 Business patterns

From known information, the Business patterns identified in Table 8-1 emerged in the HR Staffing solution.

Table 8-1 Business patterns applicable to the HR Staffing solution

	Self- service	Collaboration	Information aggregation	Extended enterprise
Provide information to stockholders.			X	
Provide an electronic guide to help find an office.	X			
A personalized job seeker/provider experience.	X			
Electronic job seeker resume posting.	X			
Electronic job provider job posting.	X			
Electronic searching of resumes and job postings.	X			
Access to profile management.	X			
Electronic apply for a job process.	X			
Integration with legacy systems for back office processing.	X			
Account executive and candidate advocate access to business process workflow information.		X		
Job seeker access to application status.	X			
Electronic fee payment submission via First National Bank.				X
Provide Instant Messaging capability between job seekers and candidate advocates.		X		

Applying the recognized Business patterns to the solution overview diagram yields the diagram shown in Figure 8-6.

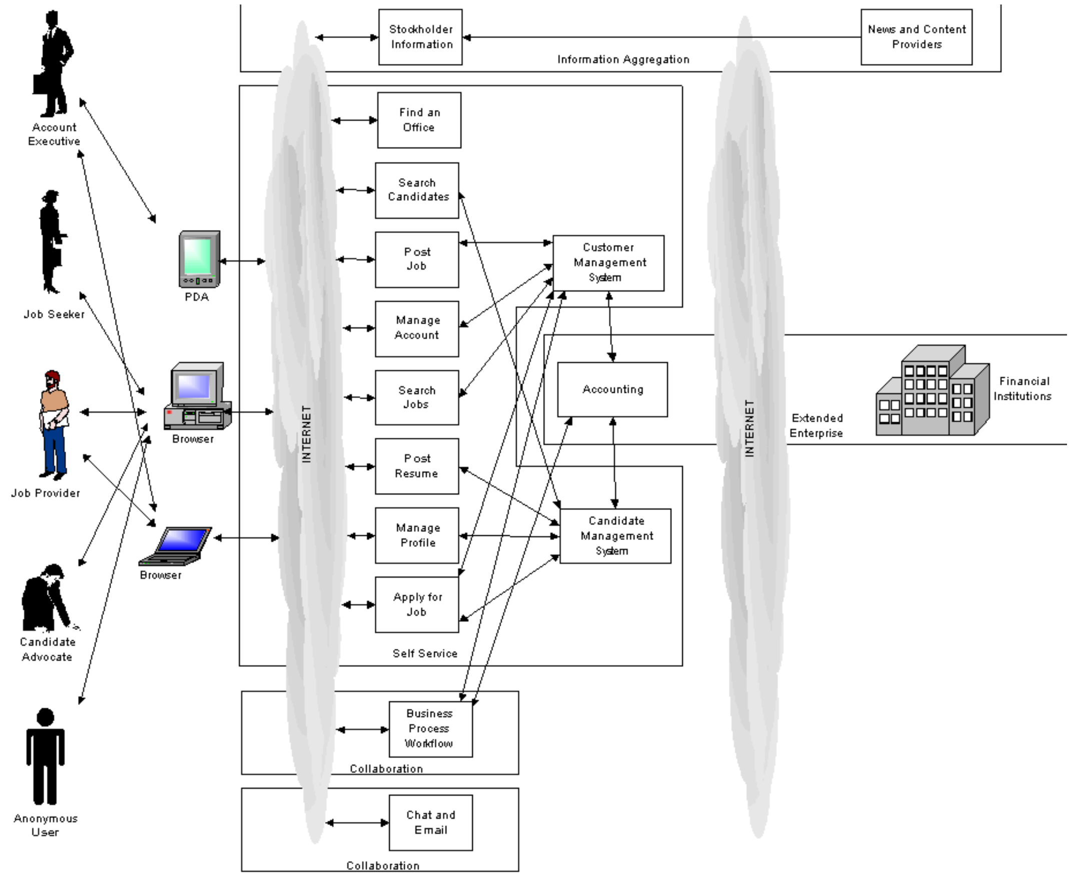


Figure 8-6 Solution overview diagram with identified Business patterns

8.5 Identify Integration patterns

We now identify the Integration patterns.

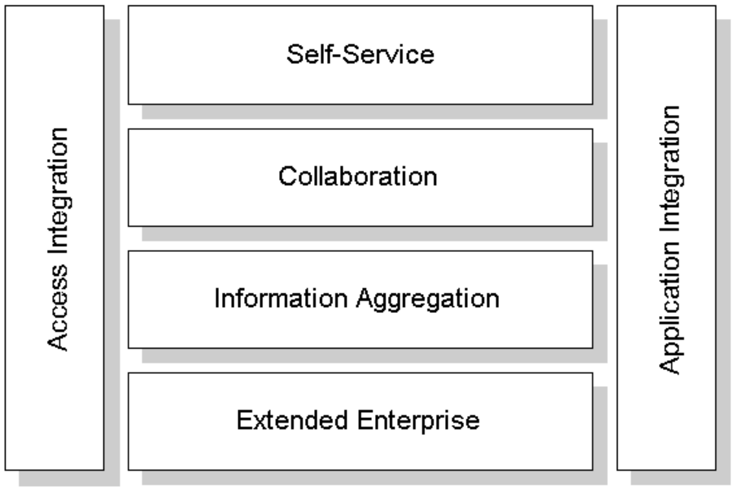


Figure 8-7 Business and Integration patterns

Access Integration will be needed to provide a seamless and consistent experience with a single sign-on. There is also the need to distinguish between different users and the roles that they adopt in the Web site. Additionally, the option of pervasive device access should not be overlooked.

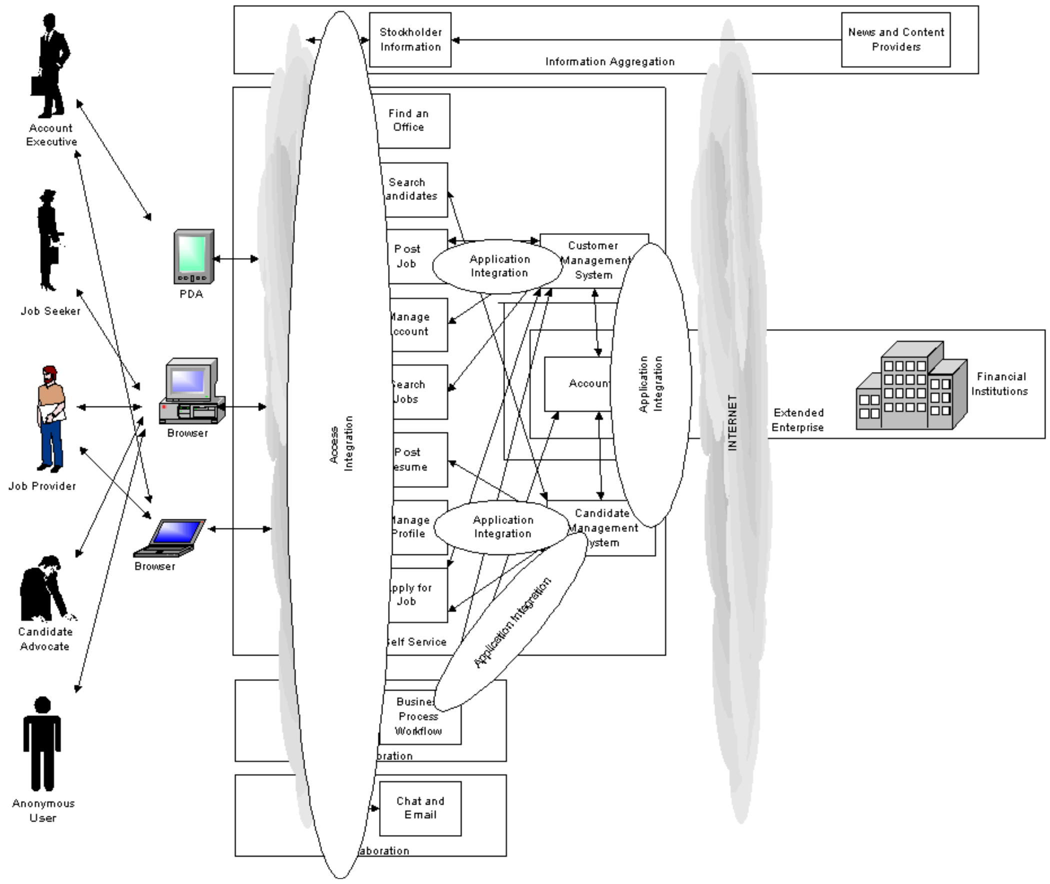


Figure 8-8 Identification of application and Access Integration patterns

8.6 Identify Composite patterns

While several patterns can be seen individually, a Composite pattern can be applied to HR Staffing. In this way, integration of several key features can be accomplished within the same solution framework.

Several Composite patterns have been identified:

- Electronic commerce
- Portal
- Account access
- Trading exchange
- Sell-side hub (Supplier)
- Buy-side hub (Purchaser)

Of these established Composite patterns, the one which most closely matches the HR Staffing situation is the *Portal* composite pattern.

Designed to facilitate many variations of similar functionality, a portal solution typically aggregates multiple information sources and applications to provide a single, seamless, and personalized access to users. A portal implementation also requires the identification of the information desired, the audience for that information, and an analysis of the usefulness of that information to fulfill the business drivers of an organization. Organizations may have one, a few, or all of these business drivers to meet their goals:

- Ease of use
- Security
- Reduce total cost of ownership
- Time to market
- Improve organizational efficiency
- Integration across multiple delivery channels
- Unified customer view across lines of business
- Support effective cross selling
- Mass customization

While a portal solution is complex in nature, the value of a Portal composite pattern is in simplification. The Portal composite pattern helps you get a jump start to your solution by predefining business and Integration patterns.

Figure 8-9 illustrates where a Portal composite pattern emerges.

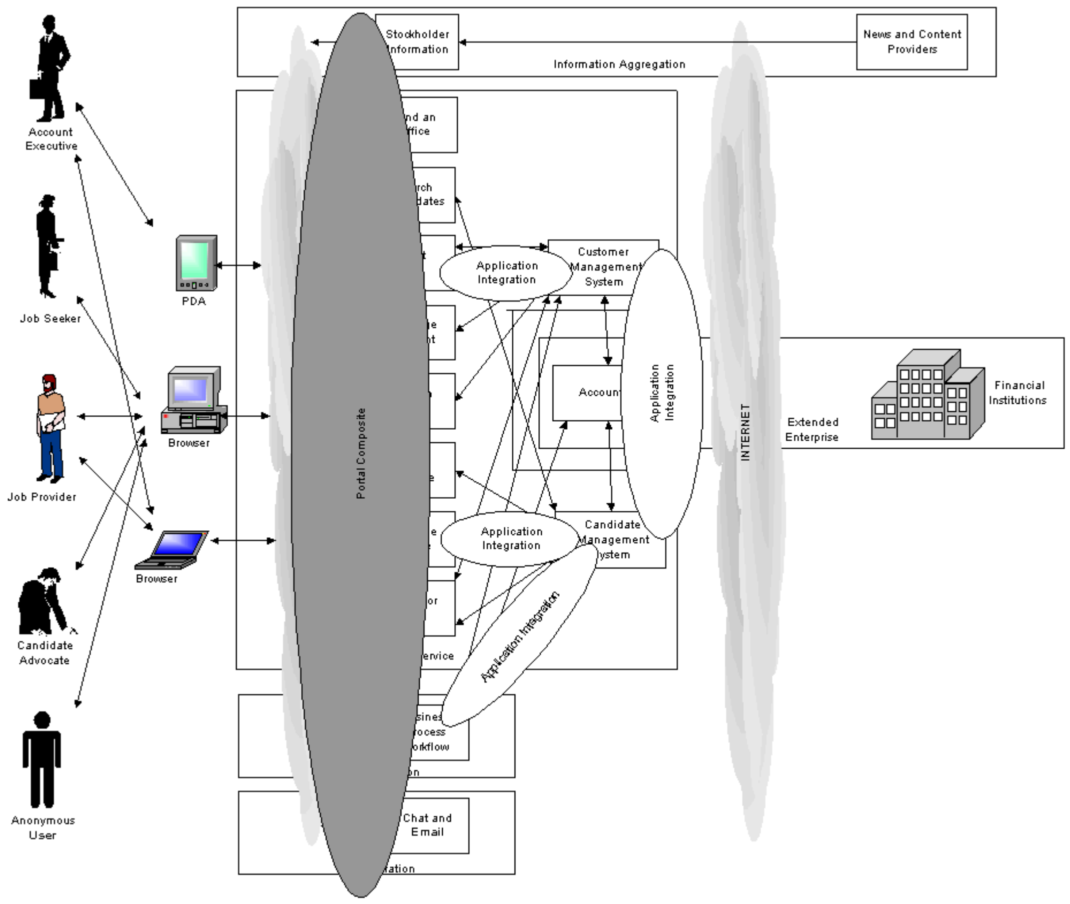


Figure 8-9 Composite pattern overlay

Note: Some might argue that the HR Staffing solution could be portrayed as a *Trading Exchange* composite model. However, the products on the HR Staffing Web site (people and jobs) are sufficiently unique that a customized solution is warranted. It would still be a viable addition to the HR Staffing solution to consider a product such as WebSphere Commerce Suite in addition to the products we discuss in this chapter. Such an approach was deemed beyond the scope of this book.

See the IBM Redbook *B2B e-commerce with WebSphere Commerce Business Edition V5.4 Patterns for e-business Series*, SG24-SG24-6194 for details about how Composite patterns can be used when developing e-commerce Web sites using WebSphere Commerce Business Edition.

8.7 Select Application patterns

8.7.1 Process analysis

Now that we have identified the high-level Composite patterns of the HR Staffing solution, let's look at the high-level interactions of the components of the system.

The interactions within the system have similarities that make them fall into two general groups, we discuss the similar interactions together in the following sections.

"Stockholder information" and "Find an office"

Figure 8-10 illustrates the "stockholder information" and "find an office" interactions.

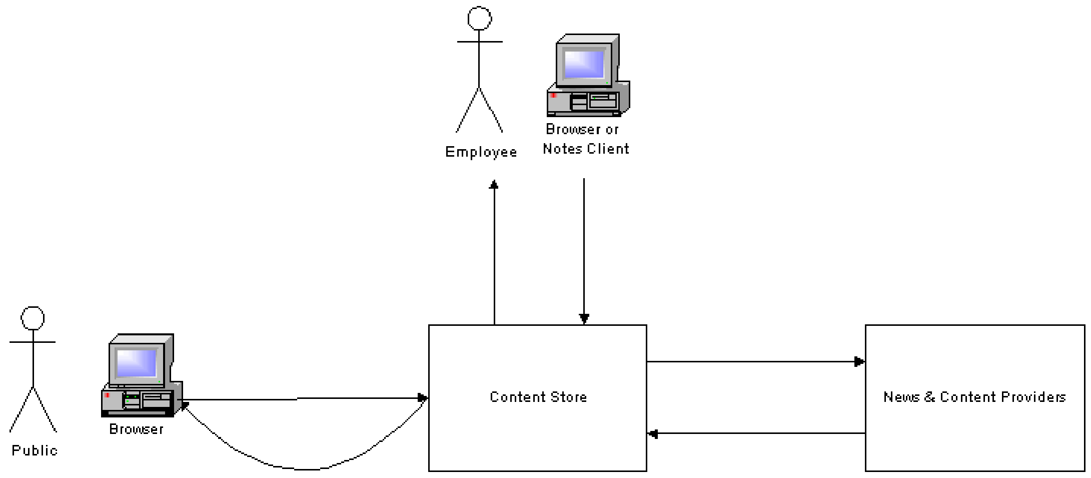


Figure 8-10 Stockholder information and Find an office interactions

Both of these processes are to be made available to the public. They are similar in that they will be stored in the content management store. Further, employees either gather or maintain the information for HR Staffing.

Search candidates, Search jobs, and so forth

Figure 8-11 represents the high-level interactions for the following processes:

- Search candidates
- Post job
- Manage account
- Search jobs
- Post resume
- Manage profile
- Apply for job

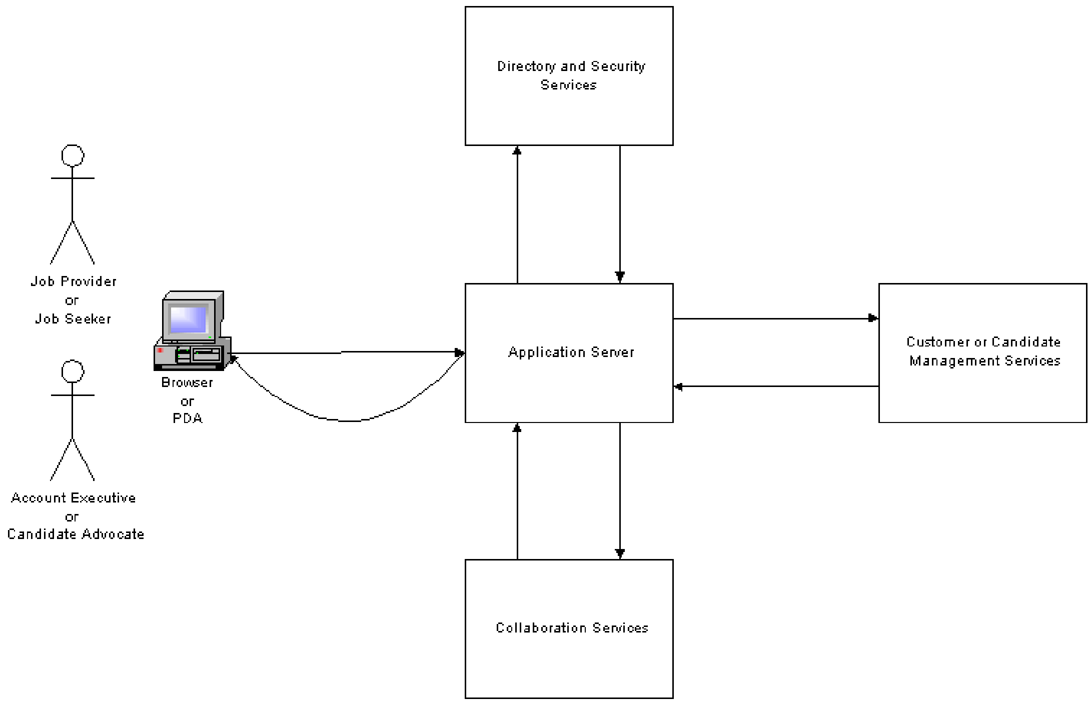

Directory and Security Services
Job Provider or Job Seeker
Browser or PDA
Application Server
Customer or Candidate Management Services
Account Executive or Candidate Advocate
Collaboration Services

Figure 8-11 Interactions for Search candidates, Search jobs, and so forth

Based on their role, the client can be either a job seeker, job provider, account executive, or candidate advocate. Each of the processes are similar in that they access the same Web site and interact with either the Customer Management System or the Candidate Management System. At different points in the process, workflow and e-mail notifications will be generated by the various systems.

8.8 Review Runtime topologies

This section begins to formulate the specific implementation of the Portal composite pattern. As always, we must be mindful of the performance, security, and scalability characteristics of a public Web site with high potential traffic.

8.8.1 Node types

This system is relatively complex. Based upon the desired functionality, we can see the need for the following nodes:

- Web server redirector

- Presentation server
- Application server
- Database server
- Directory and security server
- Firewall system
- Personalization server
- Collaboration server
- Content management (Workflow)
- Search and indexing

These nodes will be combined to create the solution.

8.9 Identify product mappings

We now map products to the nodes we previously defined. The portal composite Runtime pattern is designed to be technology agnostic. Figure 8-12 illustrates the functions the various nodes will perform; we can apply our product mapping to these nodes.

The scenario mentions that in the current state, HR Staffing has Domino deployed, serving both the Internet and intranet. The solution we recommend here realigns the product deployment based upon the recommendations in Chapter 5, “Choosing Domino-WebSphere Hybrid Runtime patterns” on page 75.

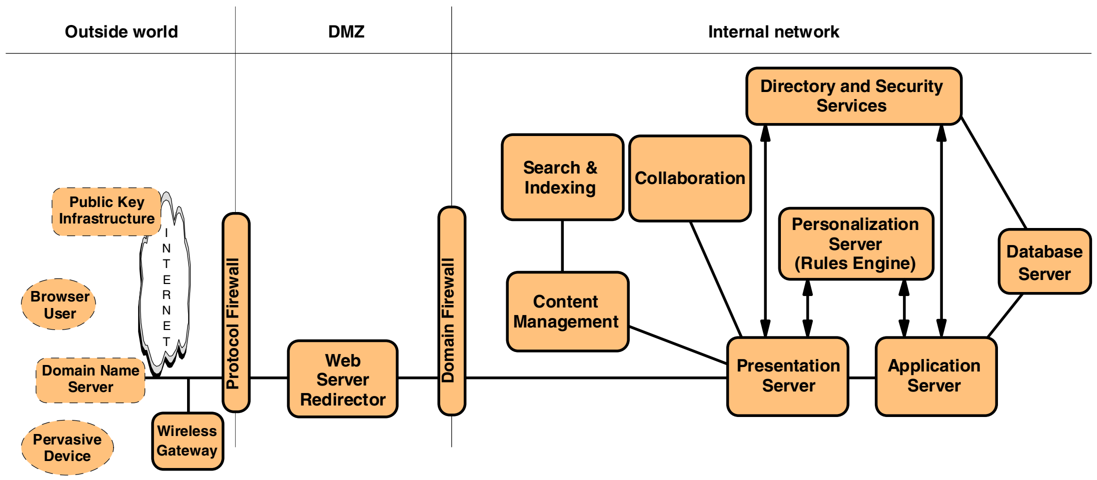


Figure 8-12 Portal composite Runtime pattern

Further inspection of the functional aspects of the nodes is warranted to align the pattern with the business scenario. The detailed functionality is shown in Figure 8-13.

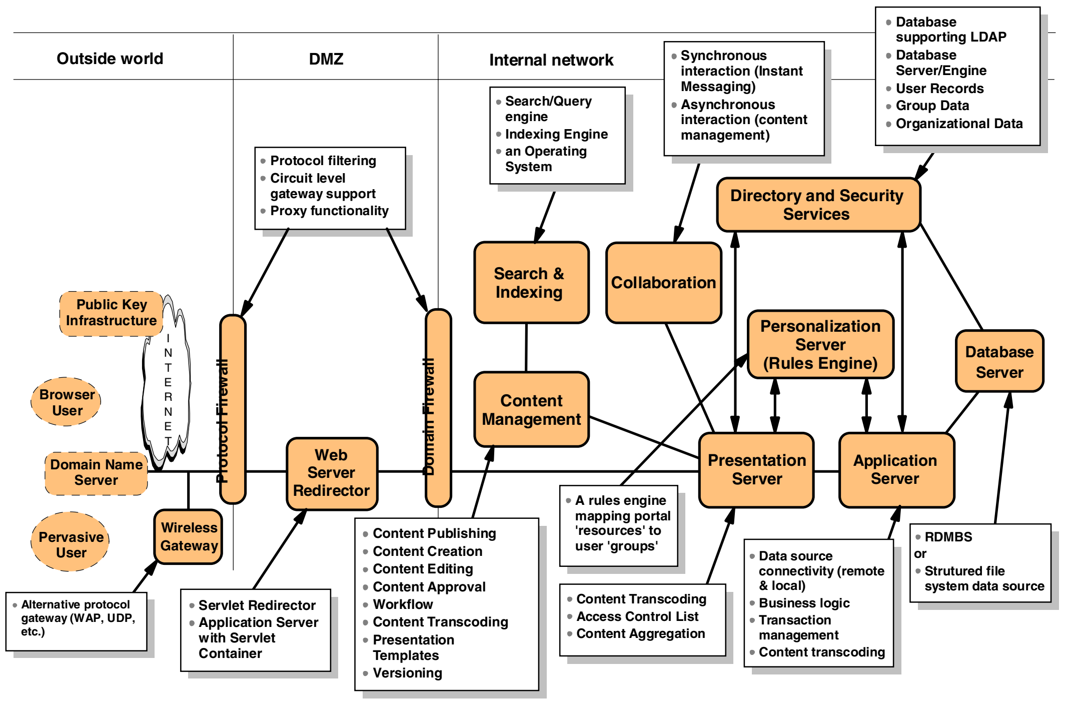


Figure 8-13 Portal composite pattern: Functional mapping

Once the Runtime pattern has been selected and functions identified, a set of products and technologies must be applied so that detailed design and implementation can occur. Figure 8-14 displays the list of products applied to each node in the Portal composite pattern.

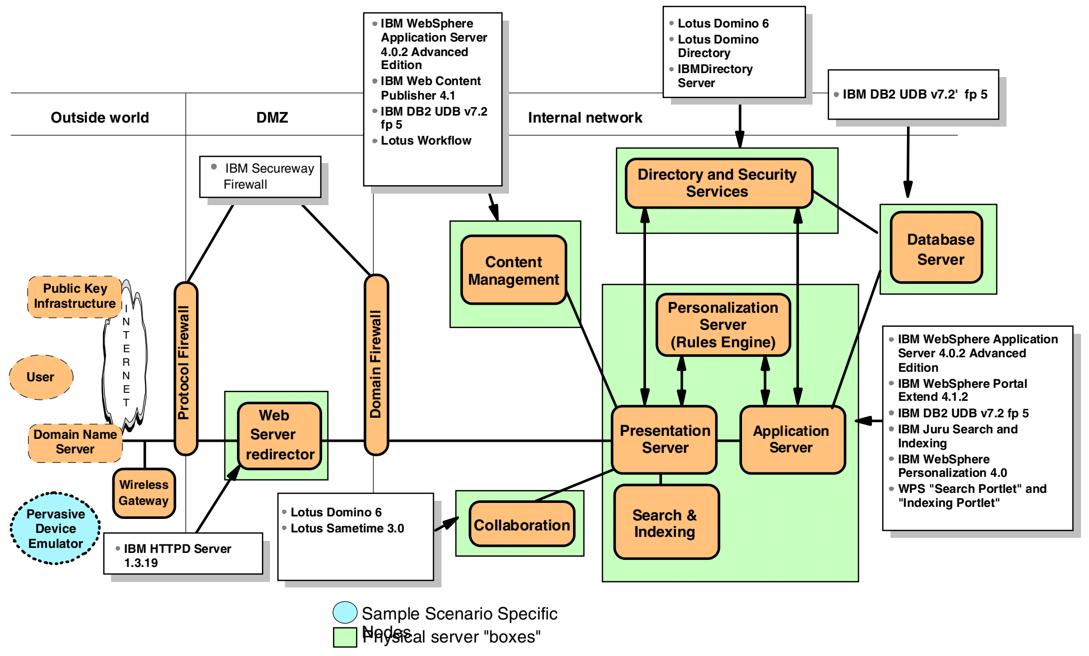


Figure 8-14 Portal composite pattern: Product mapping

Once we have the product name mapping completed, we can more closely inspect the interaction between nodes. Figure 8-15 is a view of the protocols used in the implementation.

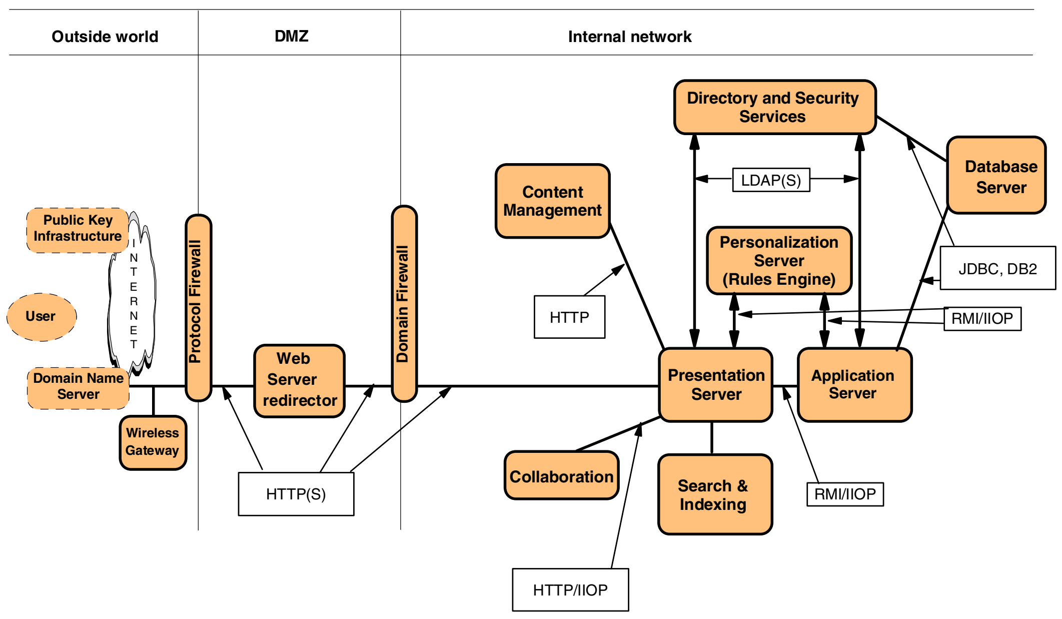


Figure 8-15 Portal composite pattern: Protocol mappings

HTTP/HTTPS: Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS) is used from the Web browser to the HTTP Server in the Web Server redirector node.

HTTP or HTTPS is also used from the WebSphere Web server plug-in in the Web server redirector node to the Web container in the presentation server node, as well as from the collaboration and content management nodes to the presentation server.

LDAP/LDAPS: The presentation and application server uses Lightweight Directory Access Protocol (LDAP) to access the LDAP server in the directory and security services node. LDAPS is the secure LDAP connection to a directory server using SSL. Since LDAP directories store essential and sensitive applications and business information, the communication can use LDAPS to be secure.

JDBC: The application server node and the directory and security services node use a Java Database Connectivity (JDBC) driver to access the database server node.

RMI/IIOP: The personalization server node uses Remote Method Invocation (RMI) over Internet Inter-Orb Protocol (IIOP) to access the EJB container in the application server node.

RMI/IIOP is also used from the presentation server node to the EJB container in the application server node.

Note: Two application servers can also communicate with each other via HTTP with SOAP using Web Services technologies.

8.9.1 Web server redirector: IBM HTTP Server powered by Apache (IHS)

The IBM HTTP Server powered by Apache (IHS) serves as the Web redirector in this scenario. Its performance capabilities makes it well suited to this role. The WebSphere HTTP plug-in will be used to redirect requests to either WebSphere or Domino as appropriate.

8.9.2 Presentation server: WebSphere Portal

WebSphere Portal provides the presentation mechanism for this scenario. Keep in mind that the WebSphere Portal is based upon the WebSphere Application Server. In this role it will extract and manage data coming from various resources, and generate end-user device output. It seamlessly provides one entry point for job seekers and job providers to the rest of the business services being offered.

For further product details refer to:

http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/portal&S_TACT=101CMM04

8.9.3 Application server: WebSphere Application Server

WebSphere Application Server provides us with the means for searching and retrieving jobs and candidates from the DB2 back-end systems. It also provides us with a readily enabled platform to connect with the Domino workflow required in several of the business processes.

Refer to 3.2.3, “Web redirector with Domino and WebSphere Application Server” on page 41 for more information about the use of the WebSphere HTTP plug-in for applications such as “Find an office.” Also, refer to 3.2.4, “WebSphere Application Server application with Domino services” on page 44 for details about the back-end calendar and e-mail support.

8.9.4 Database server: IBM DB2 Universal Database

IBM DB2 Universal Database is used for back-end data storage and retrieval. Its performance and scalability characteristics make it a market leader in relational databases.

8.9.5 Directory and security server: IBM Directory Server

The IBM Directory Server provides LDAP capabilities. This is especially useful in that both Domino and WebSphere can use its services.

8.9.6 Personalization server: IBM WebSphere Personalization 4.0

IBM WebSphere Personalization 4.0 will supply the rules engine required to personalize Web site content. It has many additional features that might be implemented by HR Staffing, such as campaign management, rule and campaign effectiveness, in addition to its profiling capabilities.

For further details, see:

http://www-3.ibm.com/software/webservers/personalization

8.9.7 Collaboration server: Lotus Domino and Lotus Sametime

Since collaboration has both synchronous and asynchronous forms, both products are warranted in the solution. Synchronous communication is handled well by Sametime, whereas Domino continues to excel in asynchronous communication. Based on the requirements of a given solution, Lotus Quickplace might also be used here in this context.

The physical implementation may vary somewhat; refer to 4.3, “Sametime collaboration topology” on page 66 for further discussion.

8.9.8 Content management (Workflow): Lotus Domino and others

For HR Staffing, the interactions between Domino and WebSphere will be largely asynchronous, driving collaborative workflow among the users of the portal solution (both employee and non-employee). In this context, Domino capably performs many roles including database, application server, content management, and more. However, Domino is not substituting in these roles.

The more classical view of content management is that of providing different types of content, publishing, and versioning control of Web content for the presentation server. While Domino could perform some of these roles, specific Domino applications would have to be purchased or written to provide the functionality. Instead, it is suggested that HR Staffing should deploy a content management solution to provide these services. Included in WebSphere Portal Server 4.1 is IBM Web Content Publisher, which can provide some of these features. For more full-featured solutions look at IBM Content Manager or a third party tool such as InterWoven TeamSite. For more information about this product, see:

`http://www.interwoven.com/products/features/overview020402/`

8.10 Summary

In order to provide a different perspective of the use patterns as they relate to WebSphere and Domino, we engineered a complex scenario and analyzed a high-level application of the pattern for e-business.

In this scenario, we:

- Identified HR Staffing’s value proposition
- Identified key business relationships, business drivers and key features
- Presented a solution overview
- Identified Business patterns
- Identified Integration patterns
- Identified Composite patterns - Portal composite pattern
- Selected Application patterns
- Reviewed Runtime topologies
- Produced product mappings

This solution could be developed further, but was limited to the current scope due to its complexity. Refer to Chapter 9, “Scalability and redundancy” on page 161 to review scalability options for the solution.

9 Scalability and redundancy

In this chapter we examine options for scalability and redundancy in a Domino/WebSphere configuration. We discuss:

- The WebSphere Edge Server
- Domino Internet Cluster Manager
- Implications of Domino together with the WebSphere Edge Server

Although this chapter deals with the options for horizontal scaling, which means adding more systems, we should point out that many large enterprises choose to scale their Web sites vertically by moving to large enterprise configurations with IBM @server zSeries or IBM @server pSeries servers. The fact that we do not cover vertical scaling here should not be taken to imply that horizontal scaling is always to be preferred. We cover horizontal scaling in greater detail because there are more complexities to be taken into account when trying to provide a single system image view when your Web site is composed of many smaller machines.

We should also point out that although this chapter deals with configuring for performance, the design and implementation of your application is also of paramount importance.

9.1 WebSphere Edge Server for Multiplatforms Version 2.0

If you are looking for a solution for improving the response time of your Web site, getting it to scale up as required and making it available to your customers all the time, then all you need is WebSphere Edge Server.

IBM WebSphere Edge Server for Multiplatforms Version 2.0 (Edge Server) is a complete solution for enabling application-aware networks. Edge Server provides an integrated solution for load balancing, static and dynamic caching, application offload, content distribution, enhanced security, and transactional quality of service—all under centralized administrative and application control.

Edge Server Version 2.0 contains many new functions not found in Version 1.0, such as application offload and content distribution. Additionally, Edge Server Version 2.0 now ships with WebSphere Application Server, Advanced Edition Single Server Version 4.0.1 for use with application offload. One hundred user licenses of Tivoli SecureWay Policy Director Version 3.8 are also included. Significant enhancements have been made to the caching proxy, load balancing, and quality of service components in Version 2.0.

Edge Server Version 2.0 provides the following functionality:

- Intelligent load balancing that directs user requests to the best server.
- Reverse proxy caching to improve the Web server response time.
- Forward proxy caching to reduce network bandwidth requirements.
- Ability to intelligently monitor the Web and application servers for better load balancing.
- Powerful rules engine and content-based routing for differentiated quality of service.
- Web content filtering and blocking if required.
- Tivoli-ready for system management.
- End-to-end SSL connectivity from browser to caching proxy and caching proxy to the Web application server.
- Content-based routing based on UserID in addition to other existing rules, to extend this service especially for POP3/IMAP servers. Also, this helps provide premium services for special customers from a larger and powerful system.

WebSphere Edge Server consists of the following modules:

- Load Balancer: A load-balancing solution for balancing requests among HTTP, FTP, and other TCP-based servers
- Caching Proxy: A Web caching and proxy server that provides better performance and enhanced proxy solutions
- Edge Services Architecture: An extension to the IBM software programming model which brings the WebSphere software platform to the edge of the network.

Detail about the use and IBM WebSphere Edge Server and its modules is beyond the scope of this redbook. For more information about WebSphere Edge Server, refer to the following IBM Redbooks and Web sites:

- IBM WebSphere Edge Server product pages:

  `http://www-3.ibm.com/software/webservers/edgeserver/`
- *Patterns for the Edge of Network,* SG24-6822
- *WebSphere Edge Server New Features and Functions in Version 2*, SG24-6511

- *WebSphere Edge Server: Working with Web Traffic Express and Network Dispatcher*, SG24-6172

9.2 Domino Internet Cluster Manager

Domino cluster support, as originally implemented, only supported Notes clients. Domino Release 5 introduced the Domino Internet Cluster Manager (ICM), which extended clustering support to Web browser clients. In Domino 6, there has been little modification.

The ICM works in the following way:

1. It uses the Domino Cluster Database, which keeps a record of which databases are available on which servers.
2. The ICM is a process which can run on a server of its own or can co-reside on a server running the normal Domino HTTP server task. If it co-resides with Domino HTTP, then it has to be configured to use a different TCP/IP port number.
3. The servers participating in the ICM cluster need to know about the ICM server, which they should do if they are part of the same domain and use a common names.nsf file.
4. All incoming browser requests are directed to servers running the ICM.
5. The ICM determines which servers hold the requested database and issues an HTTP redirect to the browser, specifying the IP address of an available server which hosts a copy of that database.
6. The browser then fetches the database from that server. Future accesses to other documents in that database go directly to the same Domino server without going through the ICM server.
7. If a link is encountered to other databases on the same server or elsewhere in the cluster, the Domino server generates a link to the ICM so that it can pick an appropriate server to honor the new request.

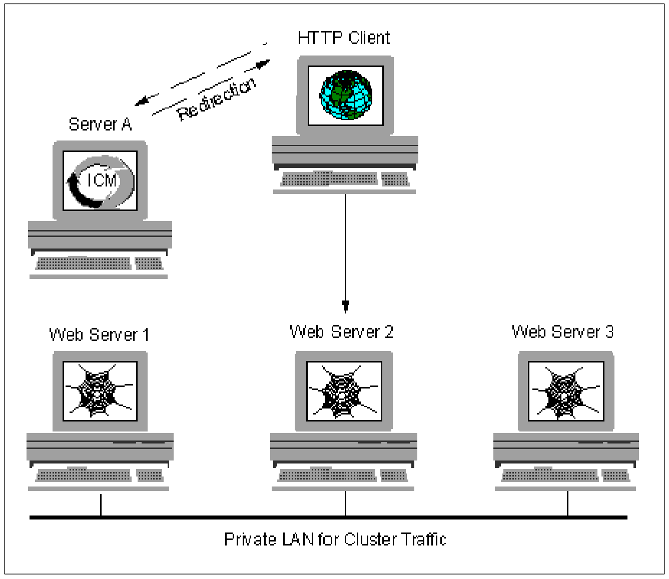


Figure 9-1 Internet Cluster Manager

Usage notes

Since a URL redirect is used, if the user looks at the URL of the page he is viewing, he will see the address of the server to which he was directed. Subsequent requests for that database continue to use that server. If that server fails, the user will explicitly have to go back to the ICM to be redirected to one of the surviving servers. If the user bookmarks a page, when he comes back to the site he will go directly to the server he was last using, regardless of its current load or, indeed, whether it is running at all.

Whether using Basic or Session authentication, the user will have to log on to each back-end server to which the ICM redirects him.

9.3 Domino authentication and WebSphere Edge Server

In this section we discuss how Domino authentication interacts with Network Dispatcher and WebSphere Edge Server, but first we review the parts of Domino authentication that are relevant to this discussion.

9.3.1 Domino authentication reviewed

If you don't set up your Domino server to use X.509 SSL certificates, you have two authentication options for Internet users:

- *Basic authentication*, as defined in RFC1945 (and RFC2068 for HTTP/1.1).
- *Session authentication*, which uses “cookies” as originally introduced by Netscape, now defined in RFC2109.

If you are thinking about building a site with several Domino servers that contain protected databases, it is important to understand the implications of using the different protection mechanisms.

Basic authentication

Basic authentication is the original, and default, method that Domino uses to control access to restricted databases. The protocol works as follows:

1. The first time you try to access a restricted resource, the Web server returns an unauthorized (401) response naming the protected *realm* and the IP address of the server containing the resource.
2. The browser then prompts you for a user name and password for that realm. You may see a panel which looks like Figure 9-2 if you use Internet Explorer.

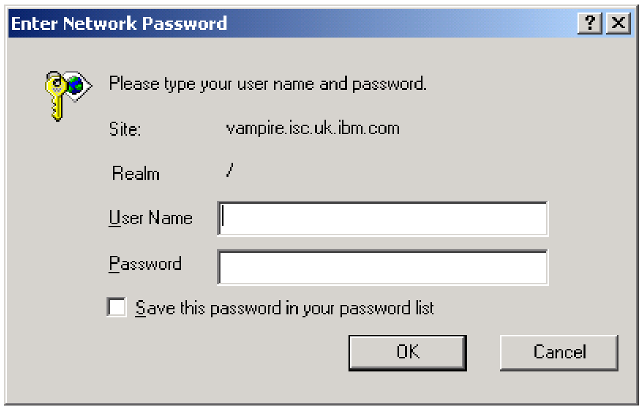

Enter Network Password
?
Please type your user name and password.
Site: vampire.isc.uk.ibm.com
Realm /
User Name
Password
Save this password in your password list
OK
Cancel

Figure 9-2 Basic authentication in Internet Explorer

Here is an example trace of an *unauthorized response* from the server that made the password dialog appear in the browser:

```
HTTP/1.1 401 Unauthorized
Server: Lotus-Domino/Release
Date: Thu, 22 Jun 2000 12:01:59 GMT
Content-Type: text/html; charset=US-ASCII
Content-Length: 297
WWW-Authenticate: Basic realm="/"
```

3. By default, the *realm* is the name of the path containing the file that you were trying to open. Domino 6 allows you to assign your own realm names to specific paths.
4. After you enter your name and password, the browser "encodes" them and re-sends the **get** request for the protected resource with the encoded user name and password in the request header.

   Here is a sample of what the Web browser sends back:

```
GET /aimetsnames.nsf HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: vampire.isc.uk.ibm.com
Proxy-Connection: Keep-Alive
Authorization: Basic RQZ3Xda3DbFjiLG1zbGV5L0FJTRVWUy9VS==
```

5. If the user name and password are valid the server returns the requested document.
6. The browser caches the user name and password (in memory) and transmits them in the header of every subsequent request to that domain and realm.

Usage notes

The realm */pathname* is considered to be a subset of the realm */*. This means that if you have authenticated against a resource in the realm /, the browser will expect the same user/password combination to work against realms like /pathname and will transmit the same user/password without further prompting. On the other hand, if you have first authenticated against the /pathname realm and then try to access /, the browser considers this to be a higher level realm and will not pass the cached user/password in the request header. If the resource in the higher level realm turns out to be protected, then you will be prompted again for your user name and password.

However, RFC1945 states "Proxies must be completely transparent regarding user agent authentication. That is, they must forward the WWW-Authenticate and authorization headers untouched, and must not cache the response to a request containing authorization."

This implies that if you are using a proxy server and want to make the best use of its cache, you should place read-only reference databases into one realm, and databases which require authentication into another named realm (which is not /). If you do have an ACL-protected database in the / realm, the browser will forward the authorization string with every subsequent request to the server and thus defeat caching.

Note: The base64 encoding of the user name and password renders them unreadable to casual users, but the encoding is easily unscrambled. So be aware (particularly if you are worried that a hacker could trace the IP traffic), that Basic authentication is not very secure, especially since the encoded user name and password are sent in the header of every request for resources in the same realm.

Session-based (cookie) authentication

Domino R5 introduced the option of using session authentication based on cookies instead of the default Basic authentication.

Session authentication is enabled under the Domino Web Engine tab in the Server document.

Session authentication offers the following advantages over Basic authentication:

- Explicit Login and Logout commands are supported (by appending ?Login or ?Logout to a Domino URL).
- A time-out can be forced after a given period of inactivity.
- A custom-tailored Login page can be provided. As this tailored Login page is a form in a Domino database, you can give it the look and feel appropriate to your Web site and it can be encrypted over SSL to protect the user/password combination from exposure to hackers.

Like the authentication header information required for the Basic authentication model, the cookie definition includes Domain and Path matching information.

Here is an example of the response from the server with the cookie:

```
HTTP/1.1 302 Found
Server: Lotus-Domino/5.0.5
Date: Thu, 22 Jun 2002 11:25:53 GMT
Location: http://azoth/aimetsnames.nsf/
Connection: close
Content-Type: text/html
Set-Cookie: DomAuthSessId=5C0A1B595FF8566F443EFB17FBC5995E; path=/
```

Note that the server has sent a 302 (relocation) response specifying the location of the protected database originally requested.

The domain defaults to the IP name of the server. The protocol defined in RFC 2109 allows the server to set the domain name to include a range of systems. For example, the domain .isc.uk.ibm.com (note the leading ".") would include all servers with a name of the form xxx.isc.uk.ibm.com. Domino does not provide an external interface to allow you to set the domain name in the response header.

The browser sends the cookie along with every HTTP request to servers whose name matches the domain definition and where the path in the request matches the path in the cookie definition.

Domino does not provide the ability for you to override the default domain name (it is always the server's fully qualified Domain Name taken from the server document) or the path, which Domino always sets to */*.

9.3.2 Load Balancer and Domino

When you have multiple identical Domino servers, fronted by Load Balancer, this works well with Basic authentication. This configuration is illustrated in Figure 9-3.

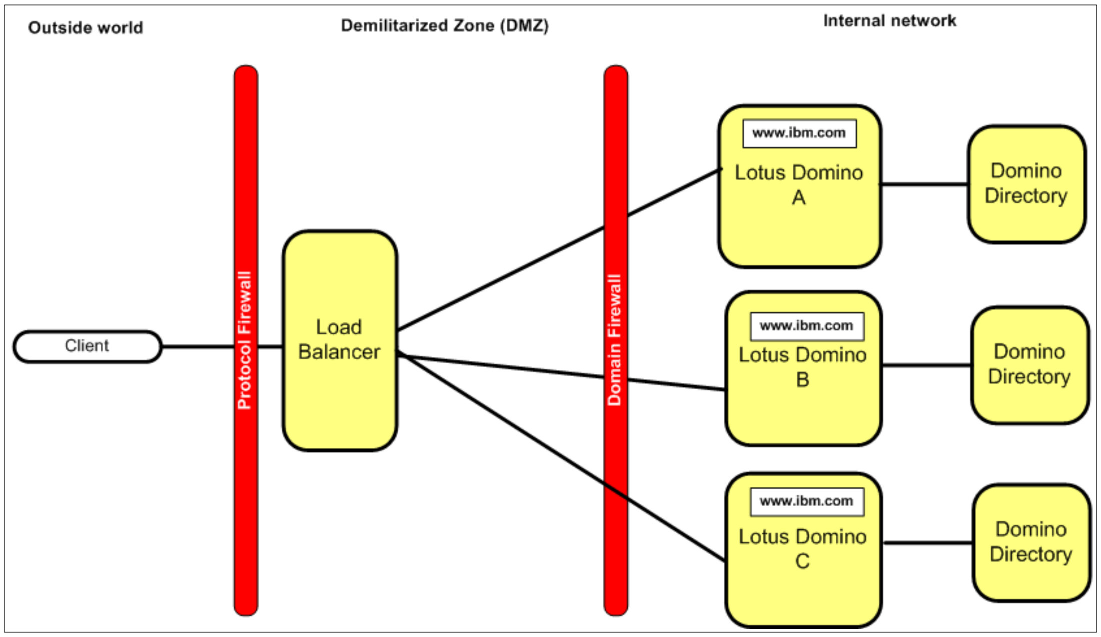


Figure 9-3 Domino and Load Balancer: Basic authentication

Once a browser user has authenticated with one of the servers, the browser will pass the validated user name and password on all subsequent requests to any of the servers; since they all have the same IP address, the browser is unaware that it is talking to several different servers. If the servers share the same Domino Directory, the same user name/password combination will be valid on each server and the user is only prompted to enter the user name and password once.

Using Domino's session (cookie) authentication, once the user has authenticated against any one server, the browser will return the cookie on every subsequent request to any of the Domino servers (since, as far as the browser knows, they are all the same server with a single IP address).

However, the Domino servers do not have any knowledge of each other's cookies. So when the browser sends back to Domino B the value of the DomAuthSessId cookie issued by Domino A, Domino B does not recognize the cookie. Domino B prompts the user to sign on again and sends a new value of DomAuthSessId back to the browser, overwriting the value set by Domino A.

If Load Balancer then routes the user back to Domino A, the user will again be re-prompted to log in.

This scenario would clearly be unacceptable.

If you want to load balance across identical Domino servers and to use cookie-based authentication, one option is to use the *sticky port* setting in Load Balancer. Requests originating from any given IP address should then always be routed to the same server (except in case of server failure), thus minimizing the likelihood of users being asked to re-logon.

Note that WebSphere advanced edition does support persistent cookies (stored in a shared DB2 repository). In a similar configuration multiple WebSphere servers would recognize each other's cookies.

9.3.3 Caching Proxy and Domino

Just like the scenario with Load Balancer, multiple Domino servers behind a reverse proxy cache (Caching Proxy) can be set up to work well with Basic authentication. You should set up realm definitions mapping all the paths to */*.

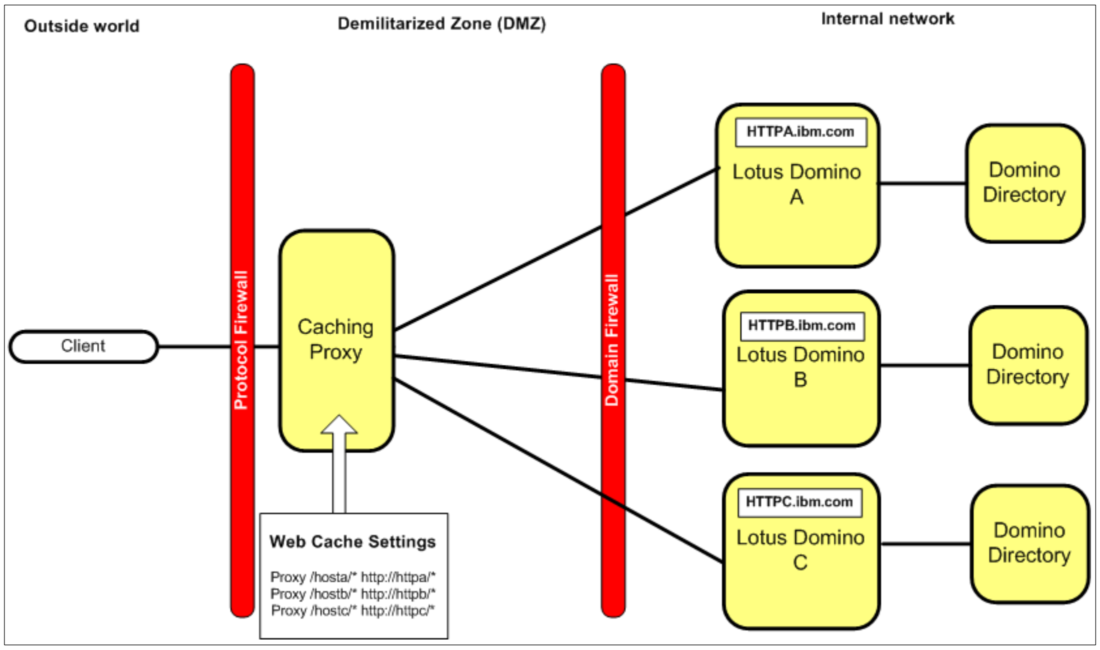


Figure 9-4 Domino and Caching proxy: Session authentication

Once a browser user has authenticated with one of the servers, the browser will pass the validated user name and password on all subsequent requests; since they all have the same IP address the browser is unaware that it is talking to several different servers. If the servers share the same Domino Directory, the same user name/password combination will be valid on each server and the user is only prompted to enter the user name and password once.

Using Domino's session (cookie) authentication, once the user has authenticated against any one server, the browser will return the cookie on every subsequent request to any of the Domino servers (since as far as the browser knows they are all the same server with a single IP address).

Like the Load Balancer scenario, Domino cookie-based authentication will be unusable in this scenario because each Domino server will be assigning its own value to the DomAuthSessId cookie and the browser does not distinguish the servers as separate.

Content-based routing

Caching Proxy in conjunction with Load Balancer can be used to provide content-based routing and load balancing across multiple groups of identical back-end servers.

Because of the problems with realm names and cookies described previously, this will work best with a single group of back-end servers if authentication is being used.

If you decide to use the caching capabilities of the Caching Proxy component of WebSphere Edge Server, you have to consider how to make sure that pages which should be cached *are* cached, and that pages which should not be cached are *not* cached.

Cache control

According to RFC 2616, by default, a response is cacheable if the requirements of the request method, request header fields, and the response status indicate that it is cacheable.

There are two ways in which the server can issue cache control directives:

- HTTP headers, whereby the server can indicate that a page is not to be cached, or if it is cached, when it should be expired from the cache.
- A <META HTTP-EQUIV="name" CONTENT="content"> tag inserted in the <HEAD> section of the page's HTML.

The Meta dictionary on `http://vancouver-webpages.com/META/` states that:

> "While HTTP-EQUIV META tag appears to work properly with Netscape Navigator, other browsers may ignore them, and they are ignored by Web proxies...Use of the equivalent HTTP header, as supported by e.g. Apache server, is more reliable and is recommended wherever possible".

HTTP headers can be generated by CGI scripts or Java programs or, with Domino, LotusScript agents. On Domino, the DSAPI can also be used to overwrite HTTP header information, but this requires programming skills that may not be present in all Domino installations.

Apache and other Web servers also allow HTTP header information to be specified using side files in a directory whose name is defined in the httpd.cnf (httpd.conf) file. The directory name defaults to .Web and the extension for the files defaults to meta according to the httpd.cnf file that is shipped with Domino on NT. However, although these entries exist in Domino's httpd.cnf file, the Domino server does not honor the definitions.

9.4 Runtime patterns for high availability and performance

Describing Runtime patterns to achieve high availability and performance is beyond the scope for this book. Instead, refer to the IBM Redbook *Patterns for the Edge of Network,* SG24-6822, which carefully details many patterns for high availability and performance. The book includes several different variants of the basic Runtime pattern to illustrate potential deployments.

A

Web services

In this appendix we describe what Web services are and discuss how they can be used in a Domino and WebSphere environment.

Topics included are the following:

- Web services introduction
- Web services standards
- Pulling it all together - the theory
- Using WebSphere Studio Application Developer wizards and example

Web services introduction

With Web services, any business system can host a Web services layer that exposes the system's underlying value in new ways. The system can, as well, consume a Web service from another system. This is true of all classes of systems: ERP or manufacturing, procurement or accounting, collaboration or workflow, messaging or interactive; enabling developers to easily and dynamically incorporate that system's core functionality in any application. This by itself isn't new: it can currently be done with system-specific APIs. What *is* new is that the functionality which the interface fronts can now be accessed without regard to platform or language.

Web services are the first common framework for dynamic, Web-based, system-to-system interaction. They take the form of modular, self-describing functions (such as a query against a shipping system or a hook into a video conferencing system) that combine existing systems, across platforms, with business processes.

Web services enable the creation of applications that navigate, discover, and use other applications, in much the same way that people navigate, discover, and use Web sites and Web-based business applications.

Web services are based entirely on accretive technology, that is, by adding capabilities to systems already in use, not by rearchitecting existing software. They eliminate most of the work and management overhead of system integration; in fact, they move network computing one giant step closer to the point at which the concept of systems integration, as we know it today, is obsolete. They enable fast, inexpensive development of dynamic e-business applications that support changing business models, and enable entirely new business models. As such, Web services technology has the potential to afford early adopters huge competitive advantages and pose a threat to the market position of organizations that fail to adapt.

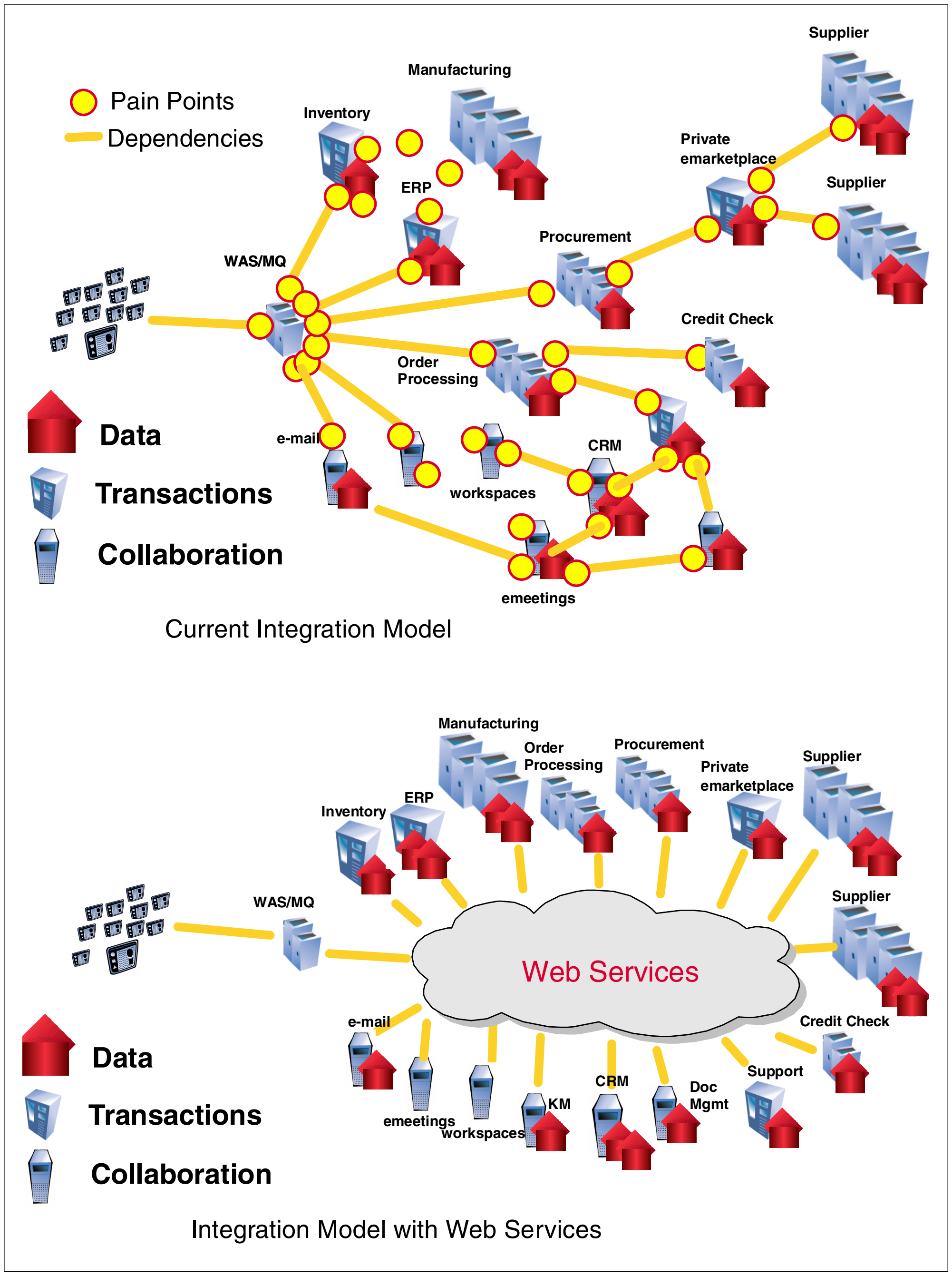


Figure A-1 The Web services model reduces the interdependencies, system-specific integration, and management overhead of dynamic Web applications

The Web services model offers its adopters significant strategic advantages, including:

- Web services leverage existing systems. As a technology designed to add capabilities to existing systems, Web services let your applications become dynamic e-business applications, simply by supporting the open technologies.
- Web services make applications simpler. The architecture eliminates the breakpoints, interdependencies, and inflexibilities of current integration models, resulting in significantly reduced development and maintenance costs.

- Web services share business resources with your partners. Because it makes process-to-process connections simple (for the first time), Web services give organizations and their partners a range of new, distributed e-business functionality.

Web services standards

Web services are enabled primarily by four open Internet technologies:

- XML (eXtensible Markup Language). XML functions as the universal language of Web services. In the same way that ASCII is universally understood by software applications, instructions or content represented by XML can be understood by any application supporting XML.
- SOAP (Simplified Object Access Protocol). SOAP is the remote procedure call (RPC) facility for Web services. With SOAP, you can send a command to another system that says, for example, “run a report against this data table, using these parameters, and return the results to me in this format.”
- WSDL (Web Services Description Language). WSDL allows Web services-enabled systems to tell each other what capabilities they have, and how to programmatically interact with them. Once a system is described in WSDL, developers can plug into the system’s programs or applications, where authorized, to create rich interactions between them.
- UDDI (Universal Description, Discovery, and Integration). UDDI is an XML-based directory, or registry, for Web services, like a “Web services Yellow Pages.” UDDI provides a place for companies to list their own Web services, and find where other companies’ Web services capabilities and resources reside.

These standards and technologies, and others rising around them, are the pieces of the Web services development model. Developers use these pieces to create distributed, modular functions that connect applications together into useful and strategic processes. These functions are not inherently transactive, but they support and trigger transactions in Web services-compliant systems.

For example, suppose one organization has a foreign currency conversion application that it wants to publish as a Web service. The organization registers the application with a Web server that supports the SOAP protocol (which runs as an HTTP extension); it then generates a WSDL description of the application which explains how it can be accessed, and then publishes the description to the UDDI directory.

Next, suppose a developer wants to make this Web service part of an application. First, the developer searches the UDDI directory and finds the Web service; the WSDL provides the details of how to call and use the Web service. Next, the developer generates code, an XML/SOAP RPC, specifying the desired functionality. The application passes the RPC to the Web service; the RPC spurs the application into action and returns a result, which in this case is a currency conversion.

The next stage in e-business evolution

Again, it is important to understand that Web services are not about changing existing applications and systems; but instead about changing the way they are exposed. The Web services model is evolutionary, not revolutionary.

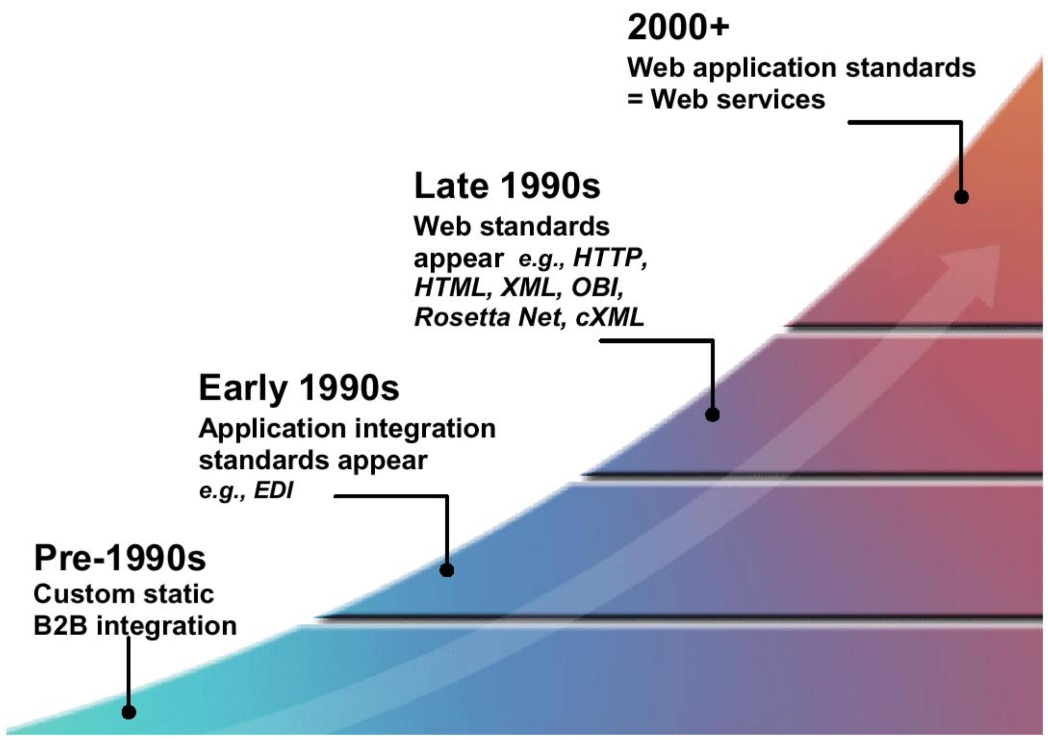

2000+
Web application standards = Web services
Late 1990s
Web standards appear e.g., HTTP, HTML, XML, OBI, Rosetta Net, cXML
Early 1990s
Application integration standards appear e.g., EDI
Pre-1990s
Custom static B2B integration

Figure A-2 The stages of integration evolution

Figure A-2 shows how systems integration and e-business have evolved in the past 10 to 12 years. Before 1990, integration took the form of data written onto magnetic tapes, and sent by courier to be loaded into another system. In the early 1990s, application standards such as EDI emerged; they were used by organizations to force value chain integration, for the purpose of increasing efficiency. In the late 1990s, Web standards appeared: HTTP, HTML, XML, OBI, Rosetta Net, and cXML, and with them e-business applications like front-to-back e-commerce and customer Self-Service. Yet still missing were standards for program-to-program integration, for example application description, dynamic discovery and binding, transaction management, and workflow.

The Web services model shifts the focus to integrating applications independent of platform, language, or data structure. Web services standards enable dynamic program-to-program integration, with shorter integration cycles and less IT investment. Developers have the power and flexibility to support new and continually changing business models.

Development, management, and business benefits

The Web services model offers significant strategic advantages in its implementation and capabilities over existing integration methods. Among these advantages are the following:

- Web services leverage existing infrastructure and application investments.

  Implicit in the Web services model is the idea that the value remains in the underlying systems, that they perform particular functions and solve specific problems. Web services technology exposes those functions in a way that makes them easy to combine and apply to new strategic ends.

  When application vendors adhere to the specifications for Web services technologies, this automates interaction with systems from a wide variety of architectural frameworks, including J2EE and Microsoft.Net, without custom coding.

- Less work to create connections; lower total cost of ownership.

  The good news about the current state of systems integration is that you can do it at all; standards support, connectors, and other technologies make it possible to integrate production systems, value chain systems, and collaboration systems throughout the extended enterprise. But it takes lots of work to make and maintain these connections, and to monitor and troubleshoot the morass of resultant breakpoints and dependencies.

  Current e-business integration methods leave multiple “pain points” where interaction can break down and careful maintenance is required.

  In contrast, the Web services model eliminates the need for much of the system-to-system integration. While developers will still need to discover the capabilities of the systems involved, the connection process is greatly simplified, and the trouble spots of integration, differing APIs and their subtleties, have been removed. Perhaps more important, integration can be reconfigured without expensive rebuilding of the system.

- Dramatic expansion of strategic enterprise and B2B e-business possibilities.

  Previous application integration standards, such as EDI, imposed constraints on the applications involved; if you wanted to do business with a company, you had to adapt the technology, and connect to and use the data in the manner prescribed by the owner. Even private e-marketplaces, while valuable, have offered to date a very constricted set of capabilities, essentially views of data preformatted by the data sources. By adopting Web services technologies, organizations make their systems available for an infinite range of new uses.

  In this way, Web services enable organizations, partners, customers, and other members of the extended enterprise to streamline their respective business processes, and create new opportunities for interaction. In a business landscape characterized by increasing customer relationship management (CRM), outsourcing, and globalization, these interactions have huge strategic value for everyone involved. Vendors can provide Web services that simplify quoting, PO processing, and virtually any other sub-processes of outsourcing. Organizations can integrate business partners and suppliers into their CRM process. Creators of private e-marketplaces can assemble and present information in ways that add significant value. Partners or internal departments can simplify collaboration by hooking into one another’s real-time chat, conferencing, or workflow systems.

To sum up: Web services enable anyone in the extended enterprise to locate an opportunity for profitable interaction, and create a process that capitalizes on the opportunity.

Putting it all together: the theory

Now that we know why Web services are so important and understand the underlying factors involved in Web standards, we can delve a bit deeper and see how that translates to the real world.

We begin by defining the roles involved in Web services, and then the different ways that Web services can be established within an existing infrastructure. This is followed by an example of how this is done using WebSphere Studio Application Developer.

Service roles

Let us first introduce the roles a Web service-enabled application requires Three roles can be identified:

- The *service provider* or *host* creates a Web service and publishes its interface and access information to the service registry.

- The *service broker* (also known as *service registry*) is responsible for making the Web service interface and implementation access information available to any potential service requestor.
- The *service requestor* or *consumer* locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its Web services.

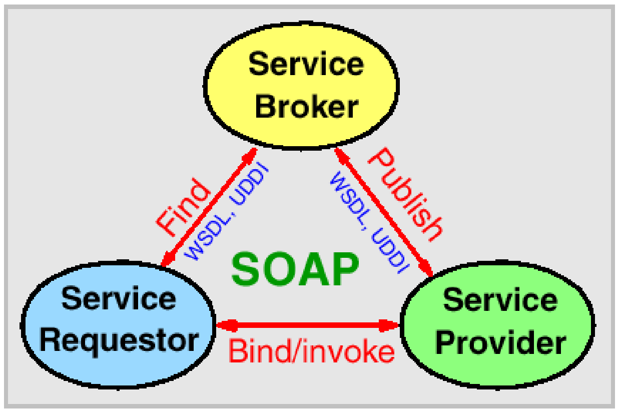


Figure A-3 Web services roles and operations

During the build cycle of any project involving Web services, certain steps have to be performed. Figure A-4 shows these steps and where the various standards are used.

.

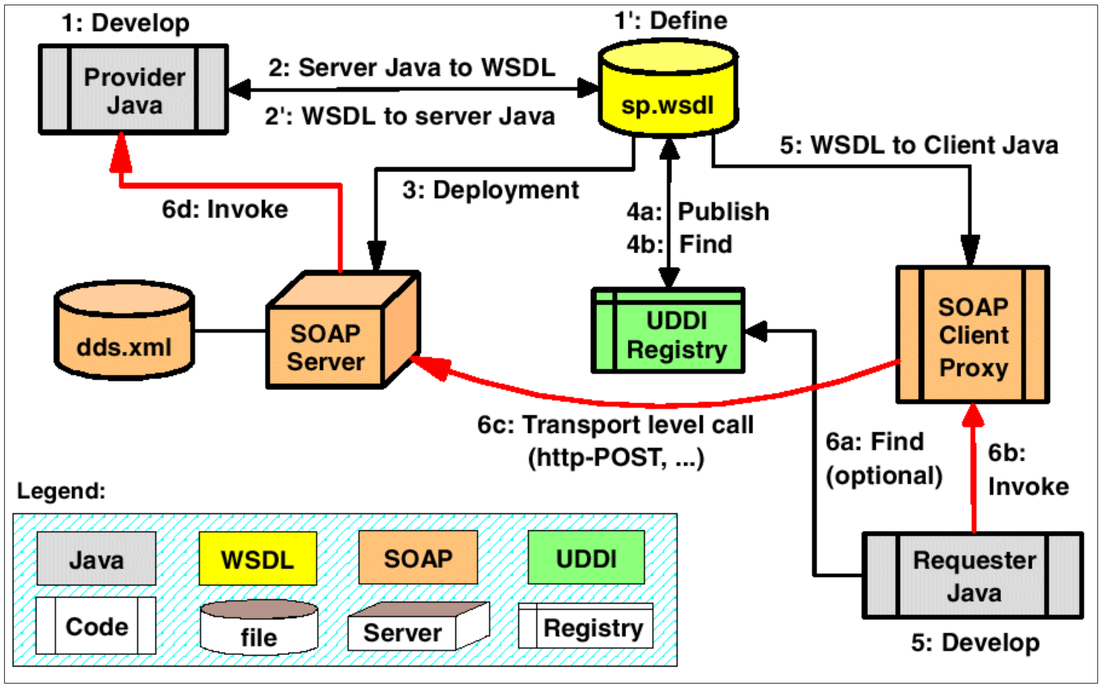


Figure A-4 Web services development and runtime information flow

The following activities are depicted in the diagram:

- Service provider-side Java development (1)
- Conversion of existing programming language artifact into WSDL interface and implementation document (2)

- Alternatively, you can start with the definition of the WSDL specification (1’) and generate server-side Java from it (2’)
- Generation of server-side SOAP deployment descriptor from WSDL specification (3)
- UDDI export and import (4)
  - Publishing, unpublishing, and update of service registrations in the UDDI registry (4a)
  - Finding service interface registrations in the UDDI registry (4b)
- Generation of client-side SOAP stub and optional generation of Web service test clients (5)
- Service invocation and execution (6)
  - Dynamic lookup of service providers in the UDDI registry (provider dynamic requestor only) (6a)
  - Service requestor-side service invocation (6b)
  - Transport level communication between requestor and provider (6c)
  - Service provider-side service invocation (6d)

Even if all specifications are human readable (XML), there is a strong need for tools supporting these development steps, and a lot of documents with overlapping content are involved. It would be cumbersome and error prone to define all these files without tools. WebSphere Studio is one of many tools that provides this automation. Domino Designer can also be used to build Web services, but it currently lacks the automation facilities to create the files required to turn functions of applications into Web services. We show you how it is possible to manually create Web services later in this appendix.

Another possibility would be to use WebSphere to front a Domino application. This would then allow a developer to use the Web services tooling within WebSphere Studio.

Development strategies for provider and requestor

Having described the roles involved in a Web services scenario and then shown the information runtime flow of a Web services transaction, it is a good time to discuss the three styles in which a Web service can be defined:

Top down: When following the top down approach, both the server-side and client-side Java code are developed from an existing WSDL specification.

Bottom up: If some server-side Java code already exists, the WSDL specification can be generated from it. The client-side Java proxy is still generated from this WSDL document.

Meet in the middle: The meet in the middle (MIM) development style is a combination of the two previous ones. There are two variants:

MIM variant 1 Some server-side Java code is already there; its interface, however, it is not fully suited to be exposed as a Web service. For example, the method signatures might contain unsupported data types. A Java wrapper is developed and used as input to the WSDL generation tools in use.

MIM variant 2 There is an existing WSDL specification for the problem domain; however, its operations, parameters, and data types do not fully match with the envisioned solution architecture. The WSDL is adapted before server-side Java is generated from it.

In the near future, we expect most real-world projects to follow the meet in the middle approach, with a strong emphasis on its bottom up elements. This is MIM variant 1, starting from and modify existing server-side Java and generating WSDL from it.

Service requestor

Web service clients (service requestor implementations, that is) have to import the WSDL interface and implementation specification of the Web service to be invoked into their environment.

Three types of requestors can be identified, which import interface and implementation information at different points in time (build time versus runtime). They are:

- Static service

  No public, private, or shared UDDI registry is involved; the service requestor obtains service interface and implementation description through a proprietary channel from the service provider (an e-mail, for example), and stores it in a local configuration file.

- Provider-dynamic

  The service requestor obtains the service interface specification from a public, private, or shared UDDI registry at build time and generates proxy code for it. The service implementation document identifying the service provider is dynamically discovered at runtime (using the same or another UDDI registry).

- Type-dynamic

  The service requestor obtains both the service interface specification and the service implementation information from a public, private, or shared UDDI registry at runtime. No proxy code is generated; the service requestor directly uses the more generic SOAP APIs to bind to the service provider and invoke the Web service.

WebSphere Studio Application Developer

This section gives an overview of the possibilities of WebSphere Studio Application Developer in the area of Web services. (For a detailed discussion of how to use the wizards, see *WebSphere Version 5 Web Services Handbook*, SG24-6891.)

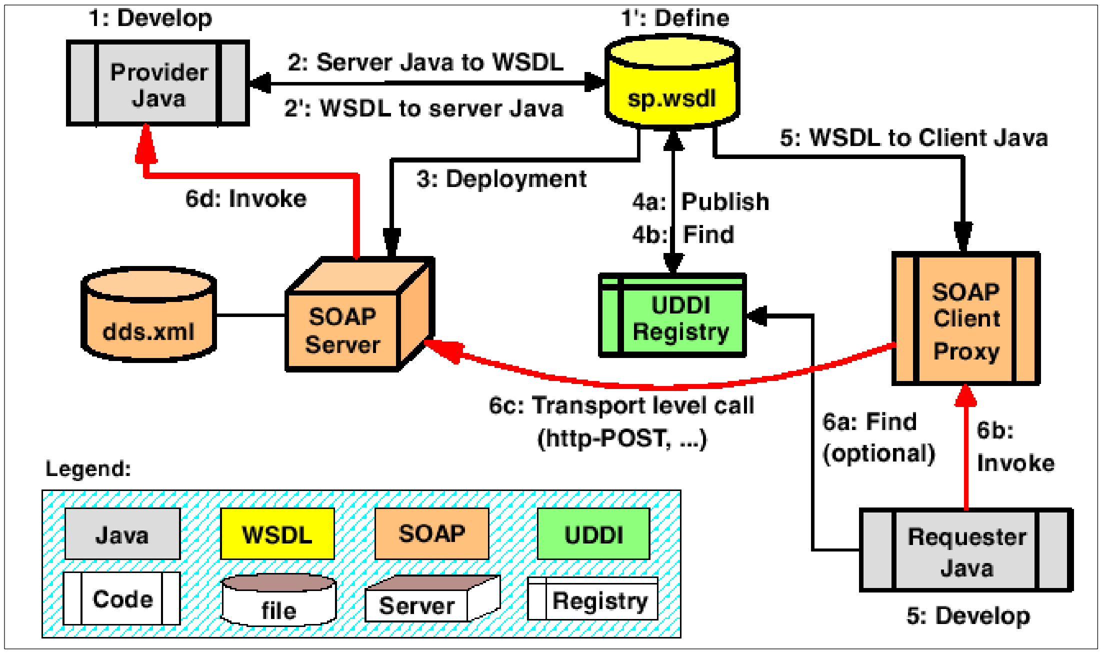


Figure A-5 Web services development and runtime information flow

Web service wizard

The Web service wizard in WebSphere Studio Application Developer starts with Step 2 in Figure A-5.

- Starting from an existing JavaBean, you specify which methods you want to expose as a Web service and the wizard generates the corresponding WSDL files (both service interface and service implementation files). The Web service wizard supports both bottom up (2) and top down (2') approaches.
- The Web service wizard generates the ISD file. The ISD file is the SOAP deployment descriptor for a particular Web service. All the ISD files in a project are composed to form the SOAP deployment descriptor, `dds.xml`, for all Web services (step 3).
- While using the Web service wizard, you have to specify the SOAP encodings. SOAP encodings tell the SOAP runtime environment how to translate from data structures constructed in Java into SOAP XML and vice versa.

  With Application Developer you can either choose literal XML encoding or SOAP encoding.
- While using the Web service wizard, you can specify to generate a Web service proxy class which conforms to step 5.

  A Web service proxy class makes it very easy to invoke a Web service. You just instantiate the class and invoke the desired method on it. The Web service proxy class itself contains the code to set the encoding styles for the input and output parameters and to make a SOAP request (step 6c).

- Another step in the wizard gives you the ability to generate and invoke a sample application.

  The sample application is a small Web application consisting of some JSPs which enable you to invoke the different methods of the Web service, and to view the results. The generated JSPs contain the code to instantiate the generated Web service proxy class and to invoke the desired methods on it (Step 6b).

  The proxy class implementation provides you with an example of how to use the Web service proxy class.
- The wizard adds two servlets to the Web application:
  - The Apache SOAP RPC router servlet
  - The Apache SOAP message router servlet

  These servlets receive the SOAP requests from the SOAP client (step 6c), invoke the Web service (for example a method on a JavaBean, step 6d) and send back a SOAP response.

  You can specify to immediately launch the sample application after the wizard is done. A default WebSphere 4.0 test environment is started. The SOAP servlets are deployed to the WebSphere 4.0 Test environment (step 3) and started. The JSP test client starts in a browser and you can test the Web service.

  To start the Web service wizard:
  - Create a Web project to contain the Web service.
  - Select the Web project and **File** → **New** → **Other**. Select **Web Services** at the left and **Web Service** at the right of the panel and click **Next** to start the wizard.

Figure A-6 shows the input and output of the Web service wizard.

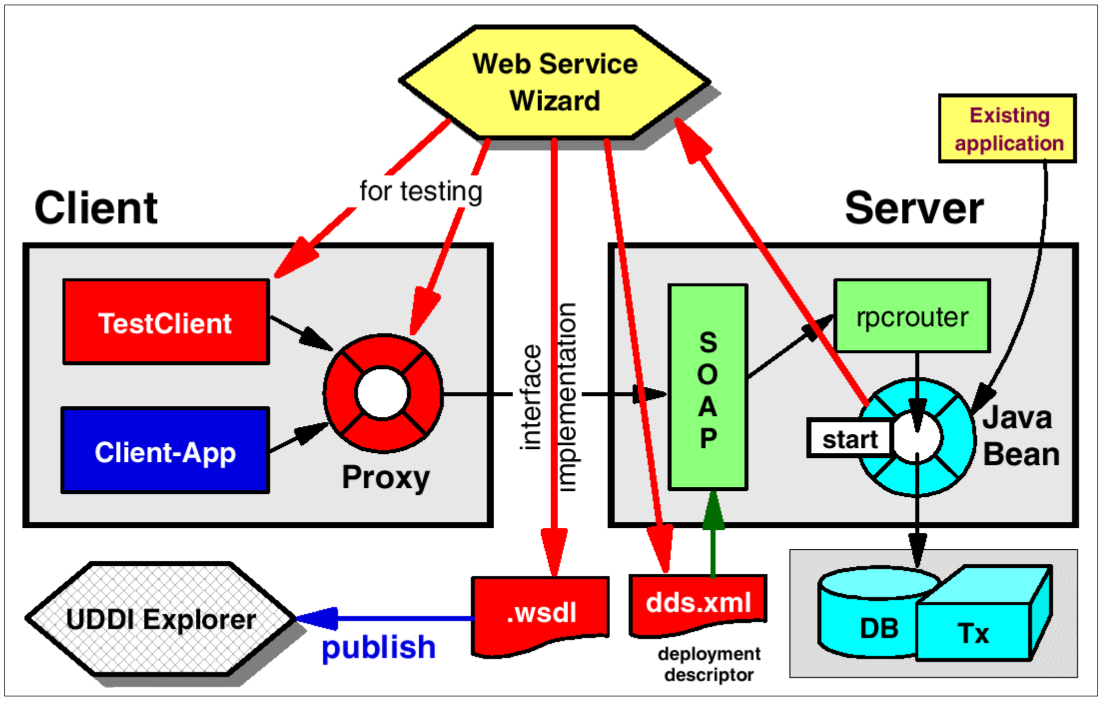


Figure A-6 The Web services wizard

The generated components include:

- WSDL interface and implementation files
- SOAP deployment descriptor (`dds.xml`)

- Client proxy for testing (JavaBean)
- Sample test client (a Web application)

Web service client wizard

The Web service client wizard gives the ability to generate a Java proxy class starting from an existing WSDL file (step 5 in Figure A-5 on page 180).

The Java proxy class encapsulates the code to make a SOAP request and to call the Web service. In fact, the Web service client wizard is part of the Web service wizard discussed previously. It also contains the ability to generate, deploy, and run a sample Web application to test your Web service.

Figure A-7 shows the input and output of the Web service client wizard.

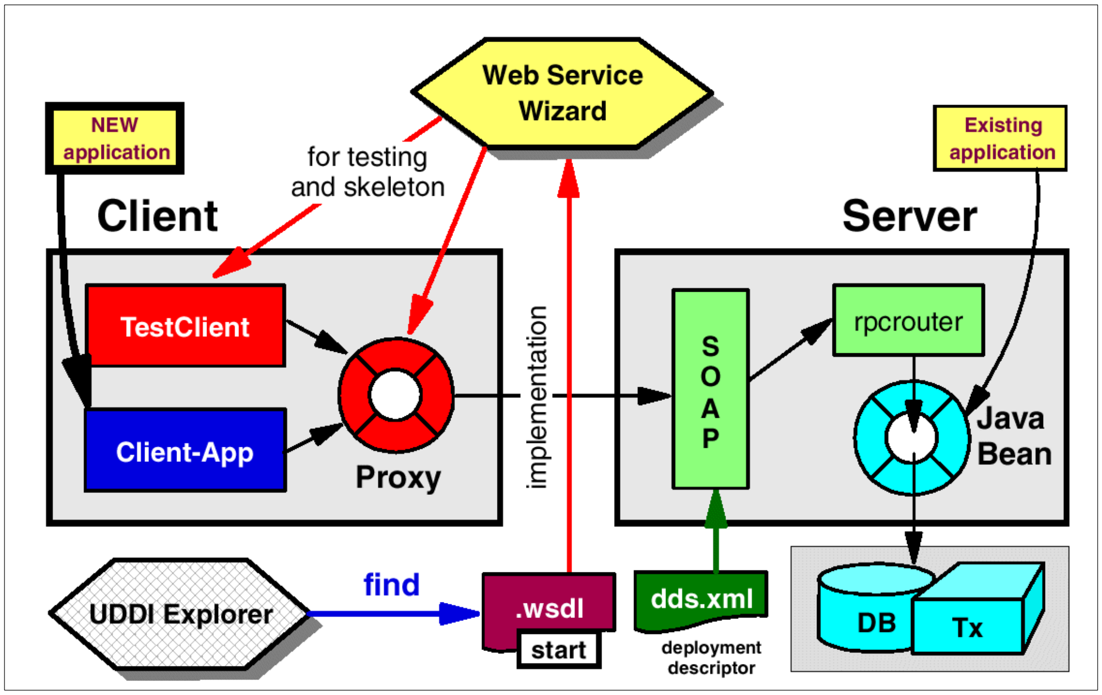


Figure A-7   Web services client wizard

The main output is the client proxy (JavaBean). The sample test client can help you in implementing the real client application.

Web service skeleton JavaBean wizard

The Web service skeleton JavaBean wizard conforms to the top down development strategy described earlier.

In this wizard you start from an existing WSDL document and generate a JavaBean. The generated JavaBean contains method definitions that conform to the operations described in your WSDL document. After running this wizard you have to implement the various methods in your skeleton bean to complete your Web service implementation.

As in the Web service wizard, this wizard also allows you to generate a Java proxy class and a sample Web application to test your Web service after you have implemented the skeleton JavaBean.

Figure A-8 shows the input and output of the Web service skeleton JavaBean wizard.

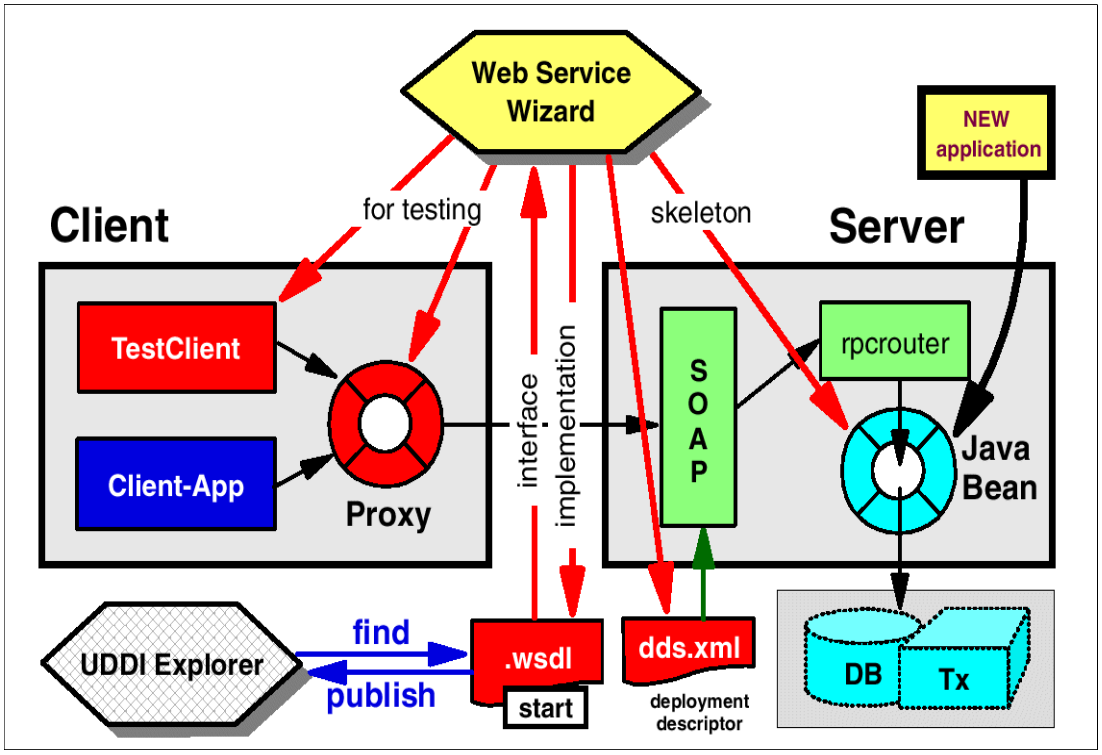


Figure A-8 Web service skeleton JavaBean wizard

For a good example, refer to the IBM Redbook, *WebSphere Version 5 Web Services Handbook*, SG24-6891, which covers Web services in detail.

B

WebSphere - J2EE compliance

This appendix provides information about the J2EE API and service levels supported by the WebSphere Application Server V3.5.2+, V4.0 and V5.0. The versions are shown in Table B-1. The service and API acronyms are defined in Table B-2 on page 186.

Table B-1 WebSphere - J2EE Compliance

	API	WebSphere V3.5.2+	WebSphere V4.0	WebSphere V5.0
J2EE Components	Servlet	2.1, 2.2	2.2	2.3
	JSP	0.9, 1.0, 1.1	1.1	1.2
	EJB	1.0	1.1	2.0
J2EE Services	JDBC	1.0, 2.0	2.0	2.0[a]
	JTA/JTS	1.0, 1.0.1, 1.1	1.1	1.1[b]
	JNDI	1.2	1.2.1	1.2.1[c]
	JAF	N/A	1.0	1.0
	XML4J	2.0.15	3.1.1	4.0
	XSL	1.0.1	2.0	2.3
	JAAS	N/A	Preview	1.0
	JCA	N/A	Preview	1.0
	JMX	N/A	N/A	1.0
J2EE Communications	RMI/IIOP	1.0	1.0	1.0[d]
	JMS	1.0[e]	1.0.1	1.0.2[f]
	JavaMail	N/A	1.1	1.2[g]
	SSL Security	N/A	Preview	2.0

	API	WebSphere V3.5.2+	WebSphere V4.0	WebSphere V5.0
Web Services	SOAP	N/A	2.2	2.3[h]
	SOAP-SEC	N/A	N/A	Preview
	UDDI	N/A	1.0.4	2.0
	WSDL	N/A	1.1	1.1

a. Includes two-phase commit (2PC) across heterogeneous databases
b. Includes support for distributed transactions
c. For EJB lookup and CosNaming
d. Includes support for JRMP
e. MQSeries native support for asynchronous messaging
f. With native provider and MQ plug-in
g. Includes Domino support
h. Includes SOAP-SEC technology preview

Table B-2 J2EE API and Services Acronyms

Acronym	Full title
EJB	Enterprise Java Bean
J2EE	Java 2 Enterprise Edition
JAAS	Java Authentication and Authorization Services
JAF	JavaBean Activation Framework
JCA	J2EE Connector Architecture
JCE	Java Cryptographic Extension
JDBC	Java Database Connectivity
JMS	Java Messaging Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
JSSE	Java Secure Socket Extension
JTA	Java Transaction Architecture
JTS	Java Transaction
RMI/IIOP	Remote Method Invocation over Internet Inter-Orb Protocol
SOAP	Simple Object Access Protocol
SSL Security	Secure Socket Layer Security. Includes JSSE and JCE
UDDI	Universal Discover, Description and Integration
WSDL	Web Services Description Language
XML4J	XML for Java
XSL	XML Stylesheet Language

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 on ordering these publications, see “How to get IBM Redbooks” on page 189.

- *WebSphere Edge Server: Working with Web Traffic Express and Network Dispatcher*, SG24-6172
- *DB2 UDB/WebSphere Performance Tuning*, SG24-6417
- *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher*, SG24-5858
- *WebSphere Edge Server New Features and Functions in Version 2*, SG24-6511
- *Enterprise Security Architecture using IBM Tivoli Security Solutions,* SG24-6014
- I*BM WebSphere V5.0 Security, WebSphere Handbook series*, SG24-6573
- *WebSphere Version 5 Web Services Handbook*, SG24-6891
- *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5: Extending Web Applications to the Pervasive World*, SG24-6233
- *Applying the Patterns for e-business to Domino and WebSphere Scenarios,* SG24-6255
- *Domino and WebSphere Together Second Edition,* SG24-5955
- *Domino Designer 6 A Developer's Handbook,* SG24-6854
- *Upgrading to Lotus Notes and Domino 6,* SG24-6889
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed,* SG24-5341
- *Lotus Sametime 2.0 Deployment Guide,* SG24-6206
- *Lotus Sametime Application Development Guide,* SG24-5651
- *Working with the Sametime Client Toolkits,* SG24-6666
- *Working eith the Sametime Community Server Toolkit,* SG24-6667
- *Design and Implement Servlets, JSPs, and EJBs for IBM WebSphere Application Server,* SG24-5754
- *IBM WebSphere Version 5.0 System Management and Configuration, WebSphere Handbook Series,* SG24-6195
- *IBM WebSphere V5.0 Security WebSphere Handbook Series,* SG24-6573
- *IBM WebSphere Application Server V5.0 System Management and Configuration: WebSphere Handbook Series,* SG24-6195-00
- *Business-to-Business Integration Using MQSeries and MQSI, Patterns for e-business Series,* SG24-6010
- *IBM Websphere V4.0 Advanced Edition Scalability and Availability*, SG24-6192
- *Patterns for the Edge of Network*, SG24-6822

- *B2B e-commerce With WebSphere Commerce Business Edition V5.4 Patterns for e-business Series*, SG24-6194
- *e-Commerce Patterns Using WebSphere Commerce Suite, Patterns for e-business Series*, SG24-6156
- *Enterprise Security Architecture using IBM Tivoli Security Solutions,* SG24-6014
- *Patterns for e-business: User-to-Business Patterns for Topology 1 and 2 using WebSphere Advanced Edition,* SG24-5864
- *Self-Service Applications using IBM WebSphere V4.0 and IBM MQSeries Integrator*, SG24-6160
- *A Portal Composite Pattern Using WebSphereV4.1,* SG24-6869
- *Access Integration Pattern using IBM WebSphere Portal Server*, SG24-6267
- *Using LDAP for Directory Integration: A Look at IBM SecureWay Directory, Active Directory, and Domino,* SG24-6163

Other resources

These publications and Web sites are also relevant as further information sources:

Books:

- *Patterns for e-business, a Strategy for Reuse,* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, George Galambos, ISBN 1-931182-02-7
- *A Pattern Language: Towns, Buildings, Construction*, by Christopher Alexander, Sara Ishikawa, Murray Silverstein, ISBN 0-1950-1919-9
- *Design Patterns,* by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, ISBN 0-2016-3361-2
- *Object-Oriented Software Engineering: A Use Case Driven Approach,* by Ivar Jacobson, ISBN 0-2015-4435-0
- *Pattern Hatching: Design Patterns Applied (Software Patterns Series)*, by John Vlissides, ISBN 0-2014-3293-5
- *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*, by Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, ISBN 0-4719-5869-7
- *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*, by Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann, ISBN 0-471606-95-2
- *The Timeless Way of Building*, by Christopher Alexander, ISBN 0-1950-2402-8
- *UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition),* by Martin Fowler, Kendall Scott, ISBN 020165783X

Web sites:

- IBM Patterrns for e-business home page includes an overview of Patterns for e-business and links to navigate through the selection and application of the patterns.

  `http://www.ibm.com/developerworks/patterns`
- IBM WebSphere Application Server Support page contains technical notes, troubleshooting help, service packs, e-fixes and more.

  `http://www.ibm.com/software/webservers/appserv/support.html`

- Lotus Domino Driver for JDBC contains downloads and links to related information

  http://www.lotus.com/products/rnext.nsf/873769A79D9C5B2285256A0800720B96/D14669BE33B75CB585256C4700659FDC?OpenDocument
- Lotus Enterprise Integration Homepage is the starting point for all information related to the Lotus Enterprise Integrator.

  http://www.lotus.com/ei
- Lotus Support contains technical notes and papers.

  http://support.lotus.com/
- "Overview of Notes/Domino Security," Lotus Developer Domain

  http://www.lotus.com/ldd/today.nsf/f01245ebfc115aaf8525661a006b86b9/ed1d81a398e0bca385256abc00105f18?OpenDocument
- "Tivoli Secureway Policy Director as the Security Mechanism for WebSphere Applications"

  http://www-4.ibm.com/software/webservers/appserv/policy_director.pdf
- "Using field encryption in applications," Lotus Developer Domain

  http://www.lotus.com/ldd/today.nsf/f01245ebfc115aaf8525661a006b86b9/24d3f7b03bcaf0c388256abb00730519?OpenDocument
- "Web Services Security (WS-Security)," IBM developerWorks

  http://www.ibm.com/developerworks/webservices/library/ws-secure/
- "WebSphere and Domino single sign-on," DeveloperToolbox Technical Magazine

  http://www.ibm.com/developerworks/library/it-0101art2/index.html

Referenced Web sites

These Web sites are also relevant as further information sources:

- IBM DeveloperWorks, the IBM resource for developers:

  http://www.ibm.com/developerworks
- Lotus Developer Domain, the premier Web site for technical information about Lotus software from IBM:

  http://www.lotus.com/developer
- Lotus Support Web site:

  http://www.ibm.com/software/lotus/support/
- WebSphere Software Platform home page:

  http://www.ibm.com/websphere

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

IBM Redbooks

Patterns: Custom Designs for Domino and WebSphere Integration

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Patterns: Custom Designs for Domino and WebSphere Integration

Harnessing the power of reusable assets

Selecting the best Patterns and products for your environment

Applying the Patterns to real-world scenarios

The Patterns for e-business are a group of proven, reusable assets that can speed the process of developing applications.

In this IBM Redbook we describe the Application Integration patterns, and how, together with one or more of the other Patterns for e-business, they form Custom designs. We first discuss the application integration methods, how IBM Lotus Domino 6 and IBM WebSphere Application Server V5 can be integrated, and then move into Hybrid Runtime patterns, where both Domino and WebSphere Application Server exist. We then expand our discussion of Hybrid Runtime patterns to include Directory integration as well as Collaboration patterns.

In the second half of the book we describe three real-life scenarios where Patterns for e-business are applied and Domino and WebSphere Application Server are part of the runtime topology. We start from the business requirements, then identify and apply the applicable Business, Application, and Runtime patterns to get to our own runtime topology. We start with a simple scenario, and then increase the complexity in the following scenarios. We discuss technology options, and provide design and development guidelines for the solutions as well.

SG24-6903-00 ISBN 0738429201