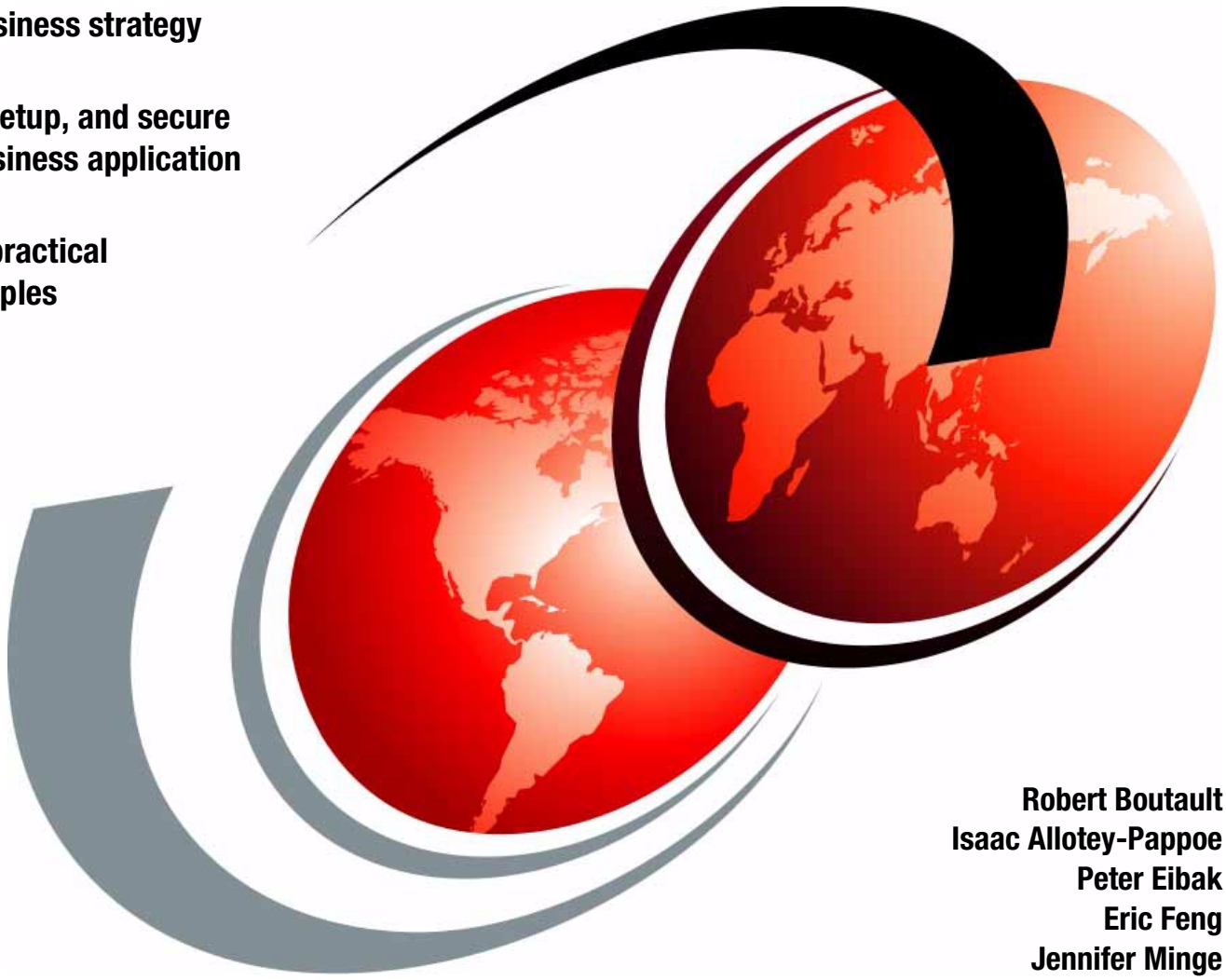
Lotus

IBM

Developing e-business Applications

Using Lotus Domino for AS/400

- Implement Domino for AS/400 in your e-business strategy
- Develop, setup, and secure your e-business application
- Discover practical new examples



Robert Boutault
Isaac Allotey-Pappoe
Peter Eibak
Eric Feng
Jennifer Minge

Redbooks

ibm.com/redbooks

International Technical Support Organization

SG24-6052-00

Developing e-business Applications Using Lotus Domino for AS/400

June 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix H, “Special notices” on page 405.

First Edition (June 2000)

This edition applies to Release 5.0.2 and later of Domino for AS/400 for use with OS/400 Version 4 Release 4 and later.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

Contents

Preface

Today, many AS/400 and non-AS/400 customers are given the project of developing an e-business application, which exploits the latest technology standards. Using Lotus Domino for AS/400 is one of the solutions that you'll want to consider.

This redbook is designed to help information systems architects and Lotus Notes developers conceive and develop an e-business application in a Domino for AS/400 environment. It defines for you the valuable role that Domino for AS/400 plays in such a project and explains the tools and processes that are involved in setting it up.

Plus, it offers you a practical understanding, through an example, of how to create an e-business application running on Domino for AS/400. The sample application demonstrates functions open to public users (Internet), functions available only to authorized business partners (extranet), and functions available only to employees (intranet). The application is used mainly through a Web browser and is protected using a large panel of security features, including SSL.

This redbook covers the following topics:

- Domino for AS/400 Web server capabilities
- The available languages, APIs, and non-programmatic tools for developing an e-business application using Domino for AS/400 (Java, JavaScript, LotusScript, DECS, Lotus Enterprise Integrator, and so on)
- The available development tools for developing an Internet application (Domino Designer R5, VisualAge for Java Professional Edition V3.0, WebSphere Studio V3.0)
- The development steps of a sample project, which highlights the new Domino 5.0 APIs and tools
- A tour through the sample application that shows the interface and how it works

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 Rochester Center.

Robert Boutault is an Advisory AS/400 Specialist at the International Technical Support Organization, Rochester Center. He writes extensively and teaches IBM classes worldwide on Domino for AS/400. Before joining the ITSO in 1998, he worked in IBM France as a Domino for AS/400 specialist in the e-business Enterprise Notes and Web Servers team and as a system engineer in Network Computing Solutions and AS/400 since 1988. He graduated from EAP, European School of Management.

Isaac Allotey-Pappoe is an IT Specialist in IBM Global Services in Sweden. He has four years of experience working with Java, Domino, Windows NT, and the AS/400 system. He holds a degree in Computer Science from the University of Science and Technology in Ghana and is currently working on his MBA from Durham University in England. He is a Certified Lotus Specialist in R4 Application

Development and Sun Certified Java Programmer. His areas of expertise include e-business, WebSphere, Java, and Domino.

Peter Eibak is an Advisory IT Specialist for IBM Denmark. He has 18 years of experience in information technology and for the past 12 years, he has specialized in the AS/400 technology field. He has worked at IBM for 18 years. He holds a masters degree in business commerce from the Danish Business School of Copenhagen. His areas of expertise include AS/400, TCP/IP, Internet, and e-business. He is one of the authors of the IBM Redbook *Cool Title About the AS/400 and the Internet*, SG24-4815.

Eric Feng is an IT Specialist for IBM China. He has three years of experience in information technology specializing in the AS/400 system. He holds a bachelors degree in computer science and a masters in software engineering from Zhongshan University in Guangzhou China. He is an IBM Certified Specialist — AS/400 Professional System Administrator. His areas of expertise include AS/400, DB2 UDB for AS/400, Domino, Java, object oriented programming, and Business Intelligence.

Jennifer Minge is a Senior Consultant that has been working for the last two years for Eagle Technology Consultants LLC, a Lotus Premier Business Partner in Atlanta, Georgia, USA. She has 16 years of experience in information technology across various platforms, and has worked with Lotus Notes for the past four years. She is a Principal Certified Lotus Professional in Application Development for R4 and R5, a Principal Certified Lotus Professional in System Administration for R4 and R5, a Microsoft MCSE, and an IBM Associate AS/400 System Operator. She holds an MBA degree in finance from San Jose State University. Her areas of expertise include client/server application development, enterprise integration, and designing and deploying Domino/Notes-based solutions. She can be contacted by e-mail at: `minge@mindspring.com`

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

Thuy Christenson
Dick Locker
Jesse Meng
Don Morrison
Pat Nickel
Terry O'Brien
Tony Perkins
Joe Peterson
IBM Programming Laboratory, Rochester

Kevin Anderson
Ted Chappell
Eagle Technology Consultants, LLC, Atlanta, Georgia, USA

Jelan Heidelberg
AS/400 Worldwide Technical Marketing, Rochester

Kurt Ruby
IBM Custom Technology Center, Rochester

Aditya Wresniyandaka
IBM Integrated Technology Services, Rochester

Marcela Adan
Bob Maatta
International Technical Support Organization, Rochester Center

Thomas Barlen
International Technical Support Organization, Raleigh Center

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks review” on page 423 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com

Part 1. Technical environment

This first part of the redbook covers the following topics:

- e-business: where Domino for AS/400 fits
- Domino for AS/400 infrastructure
- Domino for AS/400 directories and authentication management
- Domino for AS/400 SSL implementation
- Development options
- Domino development tools

Chapter 1. e-business: Where Domino for AS/400 fits

This chapter positions Domino for AS/400 as a solution to build e-business applications.

It discusses the following topics:

- The Internet and e-business
- Domino for AS/400
- Advantages of the AS/400 platform for running e-business Domino applications
- Customer requirements

1.1 Internet

The Internet is the world's largest wide area network. This network links millions of computers so that their users can exchange information.

The Internet is decentralized by design, and no one person or entity “owns” the Internet. However, many companies and educational institutions own and manage the networks that form parts of the Internet. Each Internet computer, or host, is independent.

The owners of each host choose which local services to provide to Internet users and which types of Internet services to provide to their local users.

Ten years ago, few organizations were concerned about being present on the Internet. Those who were present offered their users access to the Internet, through secured channels. At the beginning of the new millennium, the questions regarding the Internet are:

- How fast can my company get on the “net”?
- How do I:
  - Attract new customers?
  - Serve the needs of existing customers?
  - Integrate production, suppliers, and channels to bring new products and services to market?
  - Expose backend data and systems to employees, customers, and suppliers?

The millennium is about velocity and ubiquity. But security is still important, and other considerations appear.

1.1.1 The Internet phenomenon

In 1994, there was an estimated 30 million Internet users. At the beginning of 2000, the estimated number of users is almost 10 times that number, which means 300 million users all over the world. The number of Internet users is said to double every 10 months. The Internet traffic growth doubles every three months.

It is estimated there will be 128 millions Web buyers in 2002. E-commerce revenues were $43 billion in 1998. It is projected that e-commerce revenues will be $1.4 trillion in 2003.

1.1.2 e-business

e-business is the transformation of key business processes through the use of Internet technologies.

The Web is changing every aspect of our lives, but no area is undergoing as rapid and significant a change as the way businesses operate. As businesses incorporate Internet technology into their core business processes, they start to achieve real business value. Today, large and small companies are using the Web to communicate with their partners, to connect with their backend data-systems, and to transact commerce. This is e-business, where the strength and reliability of traditional information technology meet the Internet.

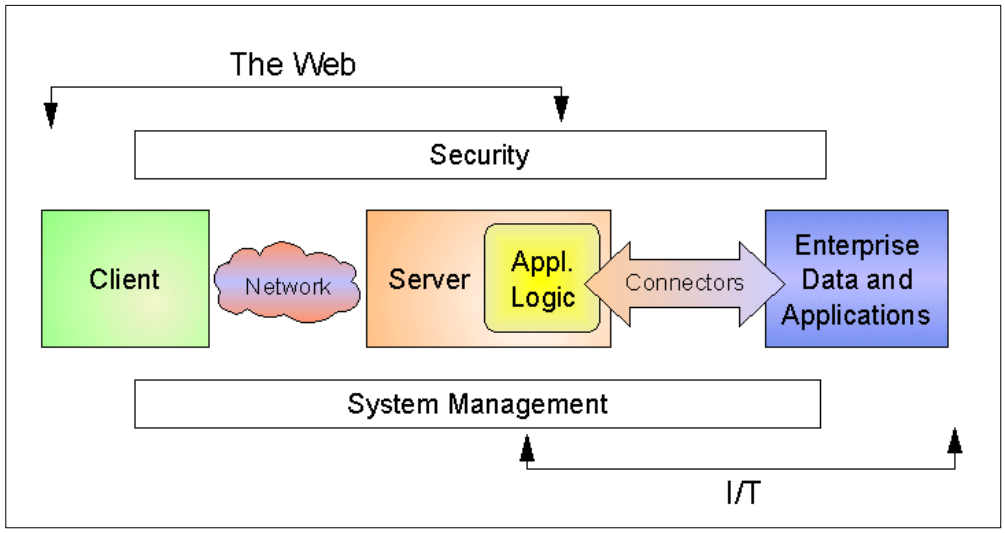


Figure 1. e-business = Web + IT

This new “Web + IT” paradigm (Figure 1) merges the standards, simplicity, and connectivity of the Internet with the core processes that are the foundation of business. The new applications are interactive, transaction-intensive, and let people do business in more meaningful ways.

An e-business is a company that can adapt to constant and continual change. To manage transitions smoothly, you have to remember two important ideas:

- Start simple, but plan to grow fast.
- Build on what you have.

e-business is about business, not technology. It’s not about re-inventing your business. It's about streamlining your current business processes to improve operating efficiencies, which in turn will strengthen the value you provide to your customers. This value cannot be generated by any other means and will give you a serious advantage over your competition.

1.1.3 Next generation e-businesses

New tendencies in business strategies and expectations are illustrated in Figure 2.

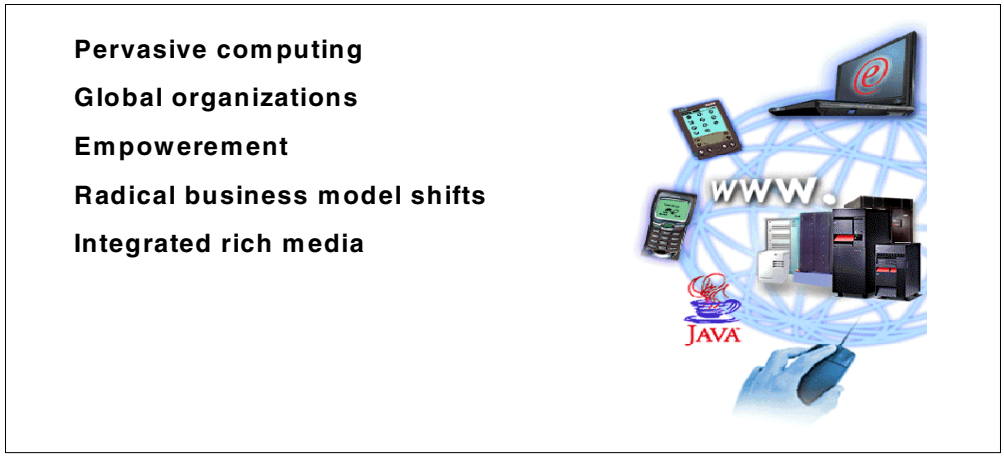


Figure 2. Next generation e-businesses

Lotus and IBM are working with customers who are at the leading edge of possibilities and who can already characterize some of the challenges that e-businesses will face in the future:

- **Pervasive computing**: There is a demand for real-time information, decisions regardless of location (home, office, mobile). The expectation is that 24-hour, seven days a week access is always on. Tools to access information are personal digital assistants or mobile digital phones. Tomorrow, they will be digital appliances that are not even dreamt of yet.
- **Global organizations**: Businesses are competing for customers across language, national, and regulatory boundaries.
- **Empowerment**: The speed of business demands local empowerment to help people act on the information they receive.
- **Radical business model shifts**: The use of information and technology is necessary to create new business services and to stay competitive.

  Fluid business relationships are required to be created or altered instantaneously.
- **Rich media**: Organizations need flexibility to seize expanded possibilities of integrated voice, video, and data, and cheap high bandwidth.

1.1.4 Lotus and IBM strategy

Together, IBM and Lotus offer certainly the industry's largest and most reliable portfolio of e-business-enabling products. The products are part of the application framework for e-business.

Lotus Domino provides you the fastest way to support Web-based relationships and processes based on messaging, workflow, and collaboration. The Domino Server Family is an integrated messaging and Web application software platform for growing companies that need to improve customer responsiveness and streamline business processes.

IBM WebSphere is a set of software products that help you develop and manage high-performance Web sites to ease the transition from simple Web publishing to advanced e-business Web applications. IBM WebSphere transforms your static Web pages to interactive, query, and transaction-intensive e-business applications.

IBM WebSphere Commerce Suite, Start and Pro Editions, help you capture the opportunities of e-commerce by providing the tools you need to quickly establish and grow your e-commerce site, as your needs dictate.

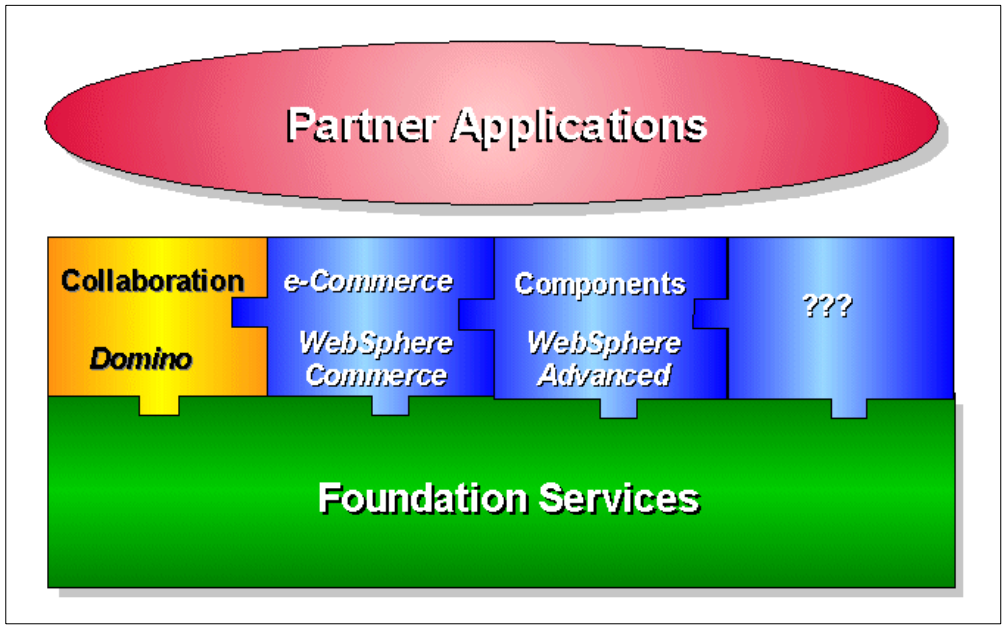


Figure 3. IBM e-business platform

This integrated portfolio of products shown Figure 3 includes two layers:

- Robust platform services (base layer)
- A set of application services offerings (top layer)

This modular architecture enables customers, and IBM and Lotus Business Partners, to rapidly deploy customized e-business applications.

Targeting emerging usage profiles

At Lotusphere 2000, Lotus outlined a strategy to provide a family of offerings that targets various emerging usage profiles in business organizations and leverages the messaging and collaboration capabilities of Lotus Domino. As part of this approach, Lotus will deliver two new offerings, iNotes and Mobile Notes, to meet the needs of users who want information delivered to match their roles, responsibilities, and working styles.

Leveraging the Lotus Domino infrastructure, iNotes will provide messaging, collaboration, mobility, and offline support to workers using any standard Web browser. The upcoming release of iNotes includes Domino Off-Line Services and iNotes Access for Microsoft Outlook. Domino Off-Line Services provides workers with easy, offline access to the Domino applications they need, thereby increasing efficiency and improving performance with minimal training. iNotes Access for Microsoft Outlook will allow Microsoft Outlook users to take advantage

of Domino's highly reliable messaging and application infrastructure, unmatched mobility, and leading collaborative applications to improve organizational effectiveness.

Mobile Notes will bring the Notes experience to intelligent handheld and wireless devices including the Palm Computing Platform and other devices such as SmartPhones. Geared toward highly mobile end users, Mobile Notes will provide offline capability for conducting collaborative e-business from any location. By leveraging upcoming releases of Lotus Mobile Services for Domino, Mobile Notes will provide access to targeted and secure information such as e-mail, calendar and directory, and other business-critical applications developed by Lotus Business Partners and enterprise Domino and Notes developers.

1.2 Domino for AS/400

This section introduces the Lotus Domino server family and then discusses on Domino for AS/400.

1.2.1 Lotus Domino

The Domino server family is an integrated messaging and Web application software platform for growing companies that need to improve customer responsiveness and streamline business processes. For those growing organizations that need to deliver secure, interactive Web applications and a very solid infrastructure for messaging and collaboration, Domino Servers are the proven choice.

Domino helps you leverage your existing investments in people, skills, tools, and backend systems. Now, you don't have to worry about tying together multiple software products for messaging, security, systems management, and data distribution and replication. Domino integrates it all.

The Domino family of servers is based on a singular architecture, so you can choose the best Domino server to meet your current needs and feel assured you have a server infrastructure with the flexibility and power to grow with you. The Domino family has three members:

- **Domino Mail Server**: Combines full support for the latest Internet mail standards with Domino's industry-leading messaging capabilities as well as calendaring functions, all in one manageable and reliable infrastructure.
- **Domino Application Server**: Includes all the functionalities of the Domino Mail Server. Plus, it is an open, secure platform optimized to deliver collaborative Web and Domino applications that integrate your enterprise systems with rapidly changing business processes.

  What is Domino Application Server? It is an unmatched rapid application development environment that:

  – Is built on a core document database “engine”
  – Is radically programmable
  – Assumes integrated security, routing, and events
  – Is complete and open
  – Is cross-platform
  – Supports all major Internet protocols, standards, and conventions
  – Is deployed to over 55 million users

- **Domino Enterprise Server**: Delivers all the functionality of the Domino Application Server. It is reinforced with partitioning and clustering for the high availability and reliability required by mission-critical applications.

1.2.1.1 Domino platform

Domino is an entire infrastructure and framework for building applications. It includes an application server, Web server, tools, processes, security and other base functions on top of which you can build applications on top. These base functions include:

- **Database**: Store and manage such non-traditional data as graphics, pictures, scanned images, scanned signatures, documents, and relational data
- **E-mail**: Mail serving and management
- **Workflow**: Managing the movement of documents and information through the organization
- **Collaboration**: Helping people work together by managing information, data, schedules, documents, communication, and business processes
- **Dynamic Web site**: Development tools and enterprise integration functions to design and build dynamic Web sites
- **Application development**: Tools, environment, security, features, and functions to develop a wide variety of applications

These base functions can be built into a wide range of applications. They are provided and managed by the Domino server and services.

Domino server and services

The core of Domino is a document database server used to run various applications. Within the broader Domino environment, there are features (messaging and replications services) and tools (Domino Designer and Java) on top of which you can build applications. The Document database server is the base of Domino, but it is complemented by the following services:

- **Database replication services**: Allow several replicas (full or partial related copies of a database) over different Domino servers. These services synchronize the replicas in a scheduled or an on-demand basis.
- **Messaging services**: Allow the sending and receiving of mail. They support many protocols, including:
  - Notes Mail
  - Post Office Protocol 3 (POP3)
  - Internet Message Access Protocol (IMAP)
  - Lightweight Directory Access Protocol (LDAP)
  - Simple Mail Transfer Protocol (SMTP)
  - Multipurpose Internet Mail Extensions (MIME)
- **Search services**: Provide the capabilities of a full text search engine to find a collection of documents based on user-defined criteria. They also maintain date view indexes.
- **Security services** include:
  - Multiple user and object access levels
  - User groups
  - Digital signatures

– Public key encryption
– Object integrity checking

- **HTTP services**: Support Hypertext Transfer Protocol (HTTP), Hypertext Markup Language (HTML), and Network News Transfer Protocol (NNTP). Domino Internet security includes support for the Secure Sockets Layer (SSL) protocol and X.509 certificates. Use your existing Internet infrastructure with Domino, and be assured that your system works with other systems based on open protocols.

Figure 4 demonstrates the richness of standards and capabilities proposed in Lotus Domino.

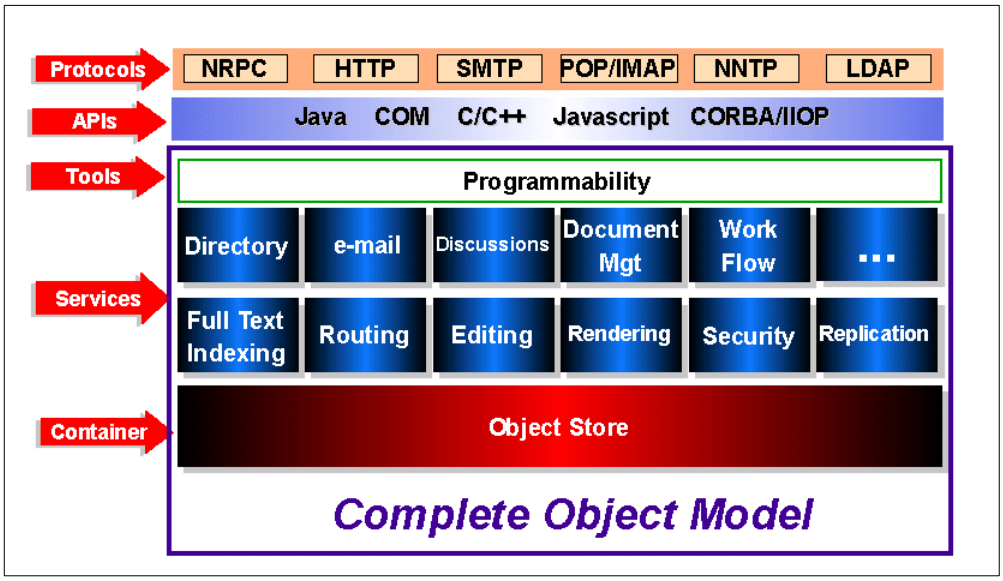


Figure 4. Lotus Domino: A platform for high value applications

1.2.1.2 Web standards implemented in Domino

Are Internet standards important? Absolutely. In fact, many people would argue that standards compliance is an absolute requirement for any software in today's marketplace. Internet standards finally smash the barrier between enterprises and between proprietary systems.

Yet the true meaning of standards gets lost amid the noise of companies claiming that standards give their products a competitive advantage. Standards are about equality, not advantages. Most Internet standards are protocol standards. They ensure that two programs of the same type can talk to each other, nothing more and nothing less. Beneath the protocols, all software is proprietary, and that's no accident. Separating protocols from software code provides a baseline for interoperability, while freeing developers to compete in the traditional areas of quality, functionality, and robustness.

Because of Domino's open design, Lotus was able to quickly achieve protocol parity with Netscape and all other vendors in the market. Internet standards-based products can interoperate fully with Domino. And developers can take advantage of Domino's rich functionality and services to create high-value, collaborative applications.

As a Web application server, Domino naturally supports a variety of Internet clients, including Web browsers, mail clients, and news readers. Unlike other Web servers, Domino also supports a wide variety of other client types based on non-Internet standards: MAPI-based mail clients, telephones, pagers, personal digital assistants (PDAs), and so on. And, of course, Domino uniquely supports the full capabilities of Lotus Notes, the world's richest integrated client.

This allows Web developers to leverage the mature groupware functionality of Notes to reach non-Notes clients through Internet protocols and formats, providing all users with greater choice and flexibility. Domino's full support for open standards and protocols means organizations are assured that new solutions from other Web vendors will fully interoperate with the Domino platform.

1.2.1.3 Internet standards and protocols

The tables presented in this section list all Internet standards and protocols supported by Domino Release 5.0.

Table 1 lists the messaging Internet standards and protocols supported by Domino.

Table 1. Messaging standards and protocols

Standard/protocol	What it is
IMAP4	Internet Message Access Protocol V4 retrieves e-mail messages from a mail server.
MAPI	Message Application Programming Interface enables different e-mail applications to share mail messages with one another.
POP3	Post Office Protocol V3 retrieves e-mail from a mail server. Most e-mail clients use POP3.
SMTP/MIME	Simple Mail Transfer Protocol is for sending e-mail messages between servers. Multipurpose Internet Mail Extensions defines formatting for non-ASCII messages so they can be sent over the Internet.
X400	A standard for addressing and transporting e-mail messages. It supports various transport mechanisms including Ethernet, X.25, TCP/IP, and dial-up lines.

Table 2 lists the communications Internet standards and protocols supported by Domino.

Table 2. Communications Internet standards and protocols

Standard/protocol	What it is
CGI	Common Gateway Interface specifies how information is transferred between Web servers and CGI programs. It is commonly used to provide interactive communication between users.
CORBA IIOP	CORBA Internet Inter ORB Protocol enables pieces of programs (called objects) to talk to one another, regardless of programming language, over the Internet. It was developed by the Object Management Group (OMB), an industry consortium.

Standard/protocol	What it is
HTML	Hypertext Markup Language is an authoring language that is used to create documents that can be displayed by Web browsers. It is a standard for how Web pages are formatted and displayed.
HTTP	Hypertext Transfer Protocol defines how messages are formatted and transmitted on the Web. It also defines the actions that Web servers and browsers should take in response to various commands.
TCP/IP	Transport Control Protocol/Internet Protocol is a set of communications protocols used to connect host computers on the Internet. It is used by the Internet and *has become the standard* for transmitting data over network.
URL	Uniform Resource Locator defines the structure of addresses on the Web. The first part of the address indicates the protocol to use (for example, ftp:// or http://) and the second part an IP address or domain name (for example, www.lotus.com).

Table 3 lists the security Internet standards and protocols supported by Domino.

Table 3. Security Internet standards and protocols

Standard/protocol	What it is
SSL 2.0 SSL 3.0	Secure Sockets Layer is a protocol for transmitting private documents through the Internet. SSL uses a private key to encrypt data that's transferred over the SSL connection.
X509	A standard for defining digital certificates. A digital certificate is an attachment to an electronic message commonly used to verify that a user sending a message is who they claim to be and to provide the receiver with the means to encode a reply.

Table 4 lists the directory Internet standards and protocols supported by Domino.

Table 4. Directory Internet standards and protocols

Standard/protocol	What it is
LDAP	Lightweight Directory Access Protocol is a set of open protocols for accessing information directories. It is based on X.500 but is simpler. Eventually, LDAP should make it possible for virtually any application on any computer to obtain directory information such as e-mail addresses and public keys.
X500	A standard that defines the hierarchical structure of global directories. It is supported by Domino for hierarchical directory naming structure.

Table 5 lists the application development Internet standards and protocols supported by Domino.

Table 5. Application development Internet standards and protocols

Standard/protocol	What it is
Java	Java is a high-level, object-oriented programming language that is well suited for use on the Web.

Standard/protocol	What it is
JavaScript	JavaScript is a scripting language for designing interactive Web sites. It has some of the object properties of Java, but is otherwise unrelated.
XML	Extensible Markup Language (XML) is a standard for creating markup languages that describe the structure and meaning of data in a document.

Table 6 lists other Internet standards and protocols supported by Domino.

Table 6. Other Internet standards and protocols

Standard/protocol	What it is
NNTP	Network News Transport Protocol is used to post, distribute, and retrieve USENET messages.
SNMP	Simple Network Management Protocol is a set of protocols for managing complex networks.

1.2.2 Domino for AS/400

With Lotus Domino for AS/400, IBM and Lotus combine the AS/400 value proposition of integration, ease-of-use, reliability, and scalability with the world's leading groupware offering. Domino for AS/400 is a full-function Lotus Domino server based on the Domino architecture.

The AS/400 implementation provides significant function and integration with OS/400. It runs as an application natively on the IBM RISC-based AS/400, exploiting 64-bit technology. It provides Domino applications with direct real-time access to the DB2 UDB for AS/400 database, supports two-way directory synchronization, coexists with other AS/400 mail services, and leverages the high availability and reliability of the AS/400 system.

Capitalizing on the integration with the AS/400 system and its stored data is the key benefit. While an external server may have some limited interaction with the AS/400 system, Domino for AS/400 maximizes the integrated access to the AS/400 system and its data by taking advantage of the AS/400 Integrated File System. Since the application is running natively on the AS/400 system, performance and scalability are also major benefits derived from this implementation. The reliability of the AS/400 system (99.97% uptime) offers a significant advantage over other Domino options. Domino for AS/400 is managed like all other AS/400 applications using the same resources and skills. And Domino for AS/400 fully exploits the 64-bit server technology of the AS/400 system to offer an additional performance improvement.

Domino for AS/400 can directly access data stored in the Integrated File System for attachment to a Notes document (in the case of PC files in the root IFS) or serving to the Web through the Domino HTTP server.

Domino for AS/400 allows direct access to DB2 UDB for AS/400 from Domino application development tools and APIs without needing an ODBC driver or a middleware communications layer, when the application code is running on the server.

DB2 UDB for AS/400 integration with Domino for AS/400 is further enhanced with support for the companion product Lotus Enterprise Integrator (LEI). Lotus Enterprise Integrator runs on the AS/400 system and allows two-way propagation of data between Domino databases and DB2 UDB for AS/400, with the exclusive feature on the AS/400 system of propagating deleted records. Lotus Enterprise Integrator on AS/400 synchronizes security between Domino databases and DB2 UDB for AS/400.

1.3 Advantages of the AS/400 platform

As a server, the AS/400 system provides reliability and scalability, which are essential in the dynamic and often unpredictable world of e-commerce. The AS/400 platform is also the perfect platform for server consolidation. As mentioned earlier, Domino for AS/400 also makes it easy to integrate the business data in DB2 UDB for AS/400 databases with the Web-enabled applications you build with Lotus Domino.

1.3.1 Reliability and availability

The AS/400 platform boasts the most robust reliability in the market, with an incredible record of 99.9+% uptime for a single AS/400 system.

Note

For information about the AS/400 system reliability, you can refer to the white paper “AS/400: 99.9+% Availability”, which can be found at:
`http://www.as400.ibm.com/whpapr/999.htm`

For customers needing better than 99.9+% system availability, AS/400 clusters are offered. They are also described in the above mentioned white paper.

Another measurement, 99.4%, has been published by the Gartner Group in the document “Platform Availability Data: Can You Spare a Minute?”, which can be read at:
`http://www.gartner.com/webletter/ibmglobal/edition2/article4/article4.html`

If a specific application should fail, the AS/400 system’s unique architecture, which logically insulates applications from one another, assures uninterrupted performance for other applications.

e-business software, such as Lotus Domino for AS/400, becomes more and more critical in the daily business operation. Customers cannot afford to have their Domino servers fail for hours. This may affect customer service levels.

Lotus Domino for AS/400 takes advantage of the reliability and availability features of the AS/400 system, such as RAID-5, mirrored disk units, and integrated backup capability.

Self-diagnostics, remote administration, and the debugging capability of the AS/400 system ensure maximum availability. AS/400 Integrated File System architecture allows for centralized backup and recovery for AS/400 applications and data, including Domino. With the Domino Enterprise Server, availability is further enhanced by the ability to run multiple partitioned servers on one physical

system and clustering to provide server failover support and dynamic load balancing.

An exclusive of Domino for AS/400 is the automatic restart of the Domino server. IBM has built a watchdog job to monitor the Domino subsystems. In the event of a failure, it automatically brings down the failing Domino server in a controlled manner and restarts it without affecting any other Domino partitions or anything else running on the system.

1.3.1.1 Domino clustering

Clustered Domino servers are a group of two to six servers that work together to provide workload balancing to improve performance and provide better scalability for your organization. All supported Domino Server platforms support clustering. Also, the members of a cluster do not have to run on the same platform. It is possible to cluster a UNIX, S/390, AS/400 system, Windows NT, and OS/2 in one cluster.

Domino clustering uses an advanced form of replication that is event-driven. This allows replicas of databases to be kept up-to-date within a second of a change. When a request for a database on a specific server cannot be performed, there is automatic failover to another server within the cluster that has a replica of the database.

The main benefits of clustering Domino servers are:

- **Failover support**: If any hardware or software problem occurs that prevents users from accessing data on a specific server, the request for data is redirected to another server within the cluster. This allows for high availability of business-critical databases.

  With servers configured in a cluster, the problems of downtime when performing required upgrades and maintenance to servers is no longer an issue. A server can be set to restricted to allow for this type of service.

- **Workload balancing**: When users request data from a server within a cluster that is heavily-used, the server passes on the request to another server within the cluster that is less used. This allows for better response times from the servers.

- **Improved performance**: Domino allows the administrators to control workload thresholds on the servers within the cluster to optimize performance. This allows for the best response times from the servers. With the monitoring databases, such as the Notes log file and Statistics & Events, in addition to the Cluster Analysis tools, the servers can be fine-tuned to further optimize performance.

- **Continuous data synchronization**: With an event-driven method of replicating databases, each server within the cluster is kept in synchronization with the latest changes, which provides the following two primary advantages:

  – If a server ends abnormally, all servers within the cluster contain the most up-to-date replica of the databases. Users can continue working with little or no time lost.

  – Each server replica is considered a backup copy of the database. If data on one server becomes irretrievable, the databases can be replaced from another server within the cluster. Information on the clustered servers is continuously updated.

There is also the option of setting up a server within the cluster that uses cluster replication to keep all databases synchronized, but that cannot be accessed by users. This server can be used for recovery in the event of total data loss on a system.

As mentioned previously, Domino clustering is available on the AS/400 system. The level of availability can be further enhanced if associating Domino clustering and AS/400 logical partitioning. This is described more in 1.3.4.2, “AS/400 Logical Partitioning” on page 18.

Placing clustered Domino servers into two different AS/400 logical partitions isolates them from software errors that affect both servers. It also provides the ability to have high-speed cluster replication between the two AS/400 partitions using the virtual OptiConnect function.

Information about clustering Domino servers on the AS/400 system is available in the “Advanced Domino Services for the AS/400” chapter of the redbook *Lotus Domino for AS/400 R5: Implementation*, SG24-5592.

1.3.2 Security

When Domino for AS/400 is installed on the AS/400 system, several actions occur in regard to security:

- AS/400 objects, such as programs (*PGM) and service programs (*SRVPGM), are put in the QNOTES library.
- Files that contain symbolic links to the programs, service programs, and other objects in the QNOTES library are created in the /QIBM/PRODDATA/LOTUS/NOTES directory in the AS/400 Integrated File System. These symbolic links provide access to the objects in the QNOTES library from the integrated file system.

  The AS/400 Integrated File System provides a directory structure similar to PC operating systems (DOS, Windows) or UNIX operating systems.
- The QNOTES user profile is created for use by Domino and Notes application programs that run on the AS/400 system.

  This user profile is intended for system functions and for integration with the underlying AS/400 security mechanisms. Therefore, it does not have a password. Individual users cannot use the QNOTES user profile to sign on the AS/400 system.

When you configure a Domino server on the AS/400 system, you create a directory for Domino data files. This directory is owned by the AS/400 user profile QNOTES, and no other AS/400 user, except the AS/400 security officer, is authorized to write in that directory. Security for the DOMSVR5 Domino server DOMSVR5 data directory, as displayed in AS/400 Operations Navigator, is shown in Figure 5 on page 16.

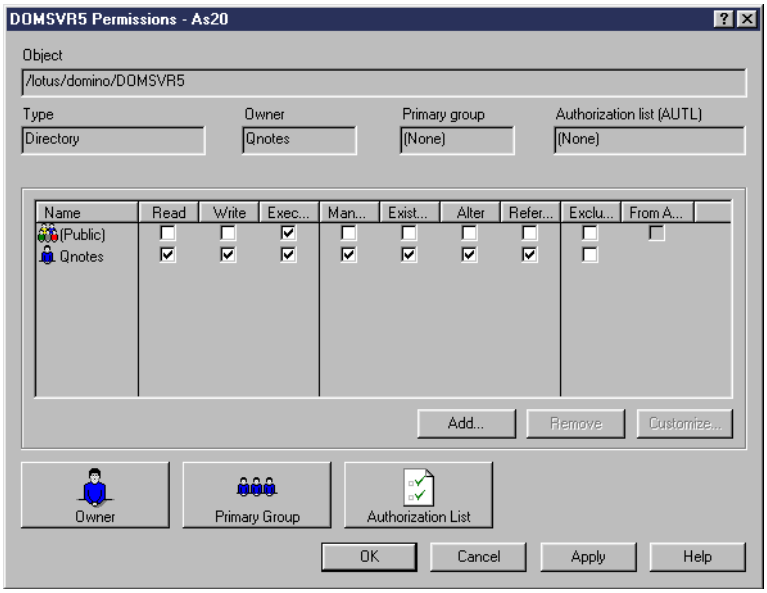


Figure 5. Domino for AS/400 server directory security

This means the Domino server environment is completely protected from direct access. This means no unauthorized AS/400 user can directly create, update, or delete a file in the Domino server data directory.

1.3.3 Scalability

Unsurpassed single footprint scalability is available with Domino for AS/400. With one administrator and one backup to perform, your administration costs are minimized as your user capacity is maximized. See Figure 6.

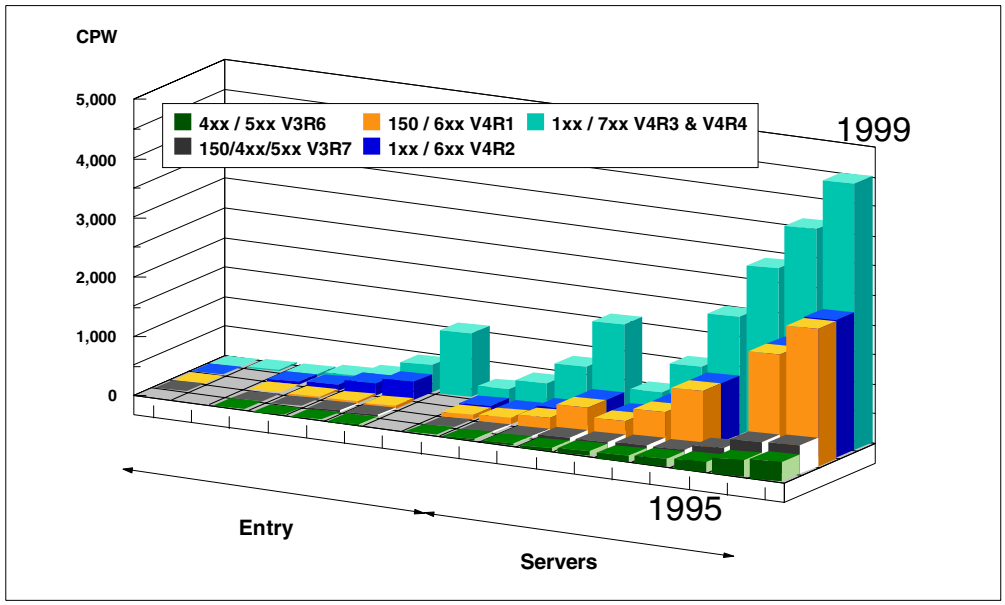


Figure 6. AS/400 scalable performance over time

The AS/400 system offers an incomparable scalability for growth of up to 230 times in the AS/400e product line with the same operating system and the same skills.

On the high-end, running on a 12-way AS/400e series model, Domino for AS/400 could accommodate 27,030 mail users, which was documented during a NotesBench audit published in November 1998.

1.3.4 Server consolidation

Scalable Domino platforms reduce the number of servers required to support an entire enterprise. The fewer the number of servers is, the lower the complexity and cost of administration and management is for an e-business or groupware solution.

Domino for AS/400 supports multiple servers on a single AS/400 system. This is referred to as partitioned Domino servers. This function allows you to operate multiple logically distinct servers, even representing different Notes domains, on the same AS/400 system.

With Domino on the AS/400 platform, you can consolidate multiple Domino servers into one box. Servers are all “under one roof”, so administration costs are a fraction of what is required for server farms.

With Domino partitioning and AS/400 subsystems, customers can safely, reliably run multiple instances of Domino doing different types of Domino applications on the same server footprint. And, on traditional AS/400 systems (which means AS/400 systems that are not Dedicated Servers for Domino), customers can combine Domino with other AS/400 work, like ERP applications.

An additional step for managing several Domino servers on one AS/400 system can be to use AS/400 Logical Partitioning, as described in 1.3.4.2, “AS/400 Logical Partitioning” on page 18.

1.3.4.1 Partitioned servers

Partitioned servers allow multiple Domino servers to run on a single computer. This feature is a part of the Lotus Domino Enterprise Server and Domino Advanced Enterprise Server for AS/400 P40/P50 licenses. It is available only for use on the AS/400 system, Windows NT, and UNIX. For Domino R5, the AS/400 system supports up to 30 partitioned servers.

Partitioning a single AS/400 system into separate servers provides the following benefits:

- Full Domino security for users of partitioned servers. Each server supports fully independent Domino security as if it were a standalone server.
- Reduced number of computers to own and administer to support independent groups of users.
- You don't need to replicate your relational databases and applications to other servers to make them accessible. All the Domino servers have access to the same relational data and traditional applications.

- When you run multiple partitions on one physical box, the partitioned servers communicate over an internal network, which does not add any network traffic to your LAN.
- The ability to tailor performance tuning characteristics on each Domino server.

Information about setting up partitioned servers is available in the “Advanced Domino Services for the AS/400” chapter of the redbook *Lotus Domino for AS/400 R5: Implementation*, SG24-5592.

1.3.4.2 AS/400 Logical Partitioning

Logical partitions are the distribution of resources (processors, memory, and I/O devices) within a single AS/400 system, making it operate as if it is two or more independent systems. Beginning with Version 4 Release 4 Modification 0 (V4R4M0) of IBM Operating System/400 (OS/400), many n-way AS/400e models (6xx, Sxx, and 7xx) support logical partitions. For more information, see *Slicing the AS/400 with Logical Partitioning*, SG24-5439.

Each logical partition operates as an independent logical system. They share a few physical system attributes (system serial number, system model, and processor feature code). All other system attributes may vary among logical partitions. Each partition has dedicated hardware (processor, memory, and I/O devices).

Unique software resources exist and operate on hardware that is assigned to each partition. These software resources include separate copies of the Licensed Internal Code, OS/400, and licensed program products. Additionally, language feature codes, security, user data, most system values, software release, and program temporary fixes (PTFs) remain unique for each logical partition.

Logical partitions can only exchange information by using standard communication mechanisms. They communicate over real, local area network (LAN), hardware wide area network (WAN), OptiConnect, or use the new virtual hardware (OptiConnect) provided with logical partitions.

An AS/400 system running Logical Partitioning (also known as LPAR) can host a different version of Domino in each partition because each partition is viewed as a separate AS/400 system. Remember, the Domino programs reside in the QSYS library in a library QNOTES. There is now one QSYS library in each AS/400 system. There is no interdependence between these libraries. They can each run the same version of Domino or one can be used for a test library for upgrades. In turn, each of these AS/400 systems can run multiple partitioned Domino servers.

LPAR is a technique that needs to be considered when a static distribution of resources (processors, memory, and I/O devices) to a Domino server is required, at least for performance reasons.

1.3.5 Total cost of ownership

The list of Domino for AS/400 values adds up to a big advantage in the Total Cost of Ownership (TCO) for AS/400 compared to a PC LAN implementation of Domino. IDC did a major TCO study in late 1998. The results showed a 32% TCO advantage for the AS/400 system over five years, based primarily on the significant differences in unscheduled downtime and in administrative costs. In late 1999, IDC updated the study for the AS/400e Dedicated Server for Domino

(DSD), smaller user populations, and a shorter time frame. The results were even more impressive, with the DSD cost advantage ranging from 43% to 51% over three years.

More information is available at: `http://www.as400.ibm.com/conslt/dsd_tco.htm`

The primary benefits of installing Domino for AS/400 include data integration and centralized management. The AS/400 system with Domino offers customers a single hardware platform to manage both their line of business applications and their e-business applications. Customers with an AS/400 Domino implementation need not invest in an additional hardware platform, operating system software, and the skills needed to maintain this environment.

Speed of deployment is a recognized AS/400 strength that also applies to Domino. Because so much is built-in and “pre-integrated” into the AS/400 system, the administrator has less work to do when the AS/400e server arrives. This allows Domino for AS/400 servers to be rolled out more quickly than other Domino servers.

Ease of administration, another important benefit of AS/400 “built for business” design and integration, means less money is spent on people managing servers.

1.4 Looking closer at customer requirements

“Deliver your business model and processes over the Internet to customers and partners is the most important project facing many companies today”. (Jim Mason, *AS/400 Experts Journal,* September 1999)

An e-business strategy is aimed at taking advantage of the pervasive nature of the Internet to provide more people access to the valuable information contained in traditional information technology (I/T) applications.

The diagram presented in Figure 1 on page 4 is generic enough to apply to any e-business solution, although not all of the building blocks illustrated may be employed. For example, a simple static-HTML Web site would only consist of the client, the network and the server. On the other hand, a solution that allowed customers with browsers to review their account status history maintained in a DB2 database would employ each of the blocks illustrated.

The diagram may become more complex when you incorporate a business-to-business scenario, where one company’s servers talk to another company’s servers across the Internet.

Customer requirements can be presented in two different ways:

- E-mail and collaboration, e-commerce, and Web application server
- Web presence, dynamic site, and transactional site

1.4.1 E-mail and collaboration, e-commerce, and Web application server

A first way to look at customer requirements is to split them in three main groups:

- E-mail and collaboration
- E-commerce
- Web application server

E-mail and collaboration requirements consist of being able to:

- Share information between groups or departments
- Overcome the obstacles of teams in different parts of the world
- Reduce the processing of paper forms and processes
- Have better control over who accesses information
- Reduce the cost of planning for projects

E-commerce requirements consist of being able to:

- Expand the base of customers
- Have hours of operation that are 24 x 7

  A 24 x 7 solution means more than simply keeping a Web server up 24 hours a day, seven days a week. If connections are made to enterprise applications or data, those systems must also support a 24 x 7 operation window.
- Provide better pre-sale and post-sale support to customers
- Make the existing Web site more dynamic in nature
- Tie the existing Web site into existing enterprise systems
- Offer payment over the Internet
- Reduce the cost of a sales transaction

Requirements for a Web application server consist of being able to:

- Tie a Web server into existing enterprise systems
- Process requests on the Web server, not just serve static pages
- Provide customized reporting to users
- Update the content of a Web site automatically
- Provide good performance and scalability
- Caching, clustering, and load balancing
- Provide session management capability

Lotus Domino is clearly the solution to answer the first requirement (e-mail and collaboration). Domino is used in many areas that are not part of the order entry process, but are still directly related to the business, like customer contacts management, customer support, leads and customers management, publishing and distribution of information (products description, contracts, and so on).

Lotus Domino can also be the solution to answer the other requirements (e-commerce and Web application server). For e-commerce, IBM WebSphere Commerce Suite is another possible solution, where IBM WebSphere can become your preferred Web application server.

1.4.2 Web Presence, dynamic site, and transactional site

Another way to define customer requirements is presented in the *AS/400e e-business Handbook*, SG24-5694, in the section “Lotus Domino for AS/400”. As detailed in the “Building e-Business Sites: Phased Approach” chapter in the handbook, customers typically go through three distinct phases when building e-business solutions. Domino can provide functionality to build and support all three types:

- Web presence
- Dynamic site
- Transactional site

The following section describes the implementation of these phases with Domino. For more information, refer to the *AS/400e e-business Handbook*, SG24-5694.

1.4.2.1 Web presence definition

As outlined earlier, Web sites serve static Web pages. The content of what the visitor sees is not changed by user interaction. In simple terms, the role of the HTTP server is to receive requests from browsers for Web pages, locate the pages, and send them to the requesters. The browser communicates with the HTTP server using URLs that contain the location of the pages that the user wants. The Domino HTTP server provides this function.

1.4.2.2 Dynamic Web site definition and architecture

A dynamic site by definition has read-only access to data on back-office business systems through a Web browser. The back-office systems we refer to and access are AS/400-based applications (ERP, home-grown, legacy, and others) and Domino applications (.nsf database). An example is a customer accessing the order status of an item purchased, or a supplier checking the inventory level at a customer warehouse.

Dynamic Web sites allow access to Domino applications through a Web browser through the Internet, intranet or extranet. The client user has read-only access through the browser to line-of-business and Domino databases and the Domino application server. Typically these applications move documents, automate processes, provide information, and respond to queries.

Domino can be used to build the following types of dynamic Web site solutions:

- Enhance and extend existing legacy applications, and provide workflow and collaboration functions not easily added with legacy development methods and tools without having to replace those core systems. At the same time, you can link this workflow and unstructured information to the wealth of business data in your DB2 UDB for AS/400 databases to enhance the value of what you've already built.
- Consolidate existing system interfaces not currently connected, and if connected, provide significant benefits.
- Automate business processes (for example, communications and e-mail) and discuss databases, movement of documents (workflow, scheduling, publication of information), distribution of documents, and access to information more accurately and with better timing.
- Rapid application development tools, templates, wizards, and data access tools allow quick development.
- Integrate existing DB2 UDB for AS/400 data into new Domino applications. Your Domino application can access your existing DB2 UDB for AS/400 customer master file for information such as a customer's name, address, and phone number.
- Provide security by supporting access controls (user authentication), digital signatures (documents and mail), encryption (communications, SSL, and mail), and protection from active content (ActiveX, JavaScript, .exe files, self-extracting .zip).

Domino for AS/400 provides this functionality with its base server functions, host integration, development tools, and messaging infrastructure. This list begins to

outline the broad scope and impact these applications can have for your business.

1.4.2.3 Transactional site definition and Architecture

A transactional site has add, delete, and edit access to data on back-office business systems through a Web browser. The back-office systems we refer to and access are AS/400-based applications (ERP, home-grown, legacy, and so on) and Domino applications (.nsf database). Some examples are a customer accessing and updating their personal information, a distributor placing an order for an out-of-stock item with their supplier, or a customer placing an online order.

Transactional sites can be divided into three main categories for analysis purposes:

- **Business-to-business**: This type of transactional site provides external supply chain (partners, vendors) add, delete, and edit access to a company's databases. Some examples include ordering products, filing sales forecasts, and accessing and updating account data. Applications on the host system process the transaction, update data, and trigger other events depending on the application design. This type of solution typically has some form of user authentication, for example: login and ID, a forms-based information request or query function that is user defined, and a forms-based data entry function for the transaction detail.
- **Business-to-consumer**: This type of transactional site provides consumers with add, delete, and edit access to a company's databases. This is different from a dynamic site where users can only view the database. Transactional sites have add, delete, and edit functions. Customers can update their personal information, be added to databases, complete information forms, application forms, and so forth. This information is then processed. The corresponding applications on the host manage the transaction. As with business-to-business solutions, this typically has some form of user authentication, for example: login and ID, a forms-based information request or query function which is user defined, and a forms-based data entry function for transaction detail.
- **E-commerce (e-store):** The e-commerce implementation involves actually transacting commerce and purchasing products or services. Business-to-business and business-to-customer sites provide data and information and allow add, delete, and edit functions, but there is no shopping. E-stores, as they are commonly referred to, can have shopping carts, shipping functions, product catalogues, order tracking, some form of back office integration, order processing, and accept credit payments, among other possible functions.

  The store model, shown in Figure 7, graphically presents the steps that customers follow in the shopping process.

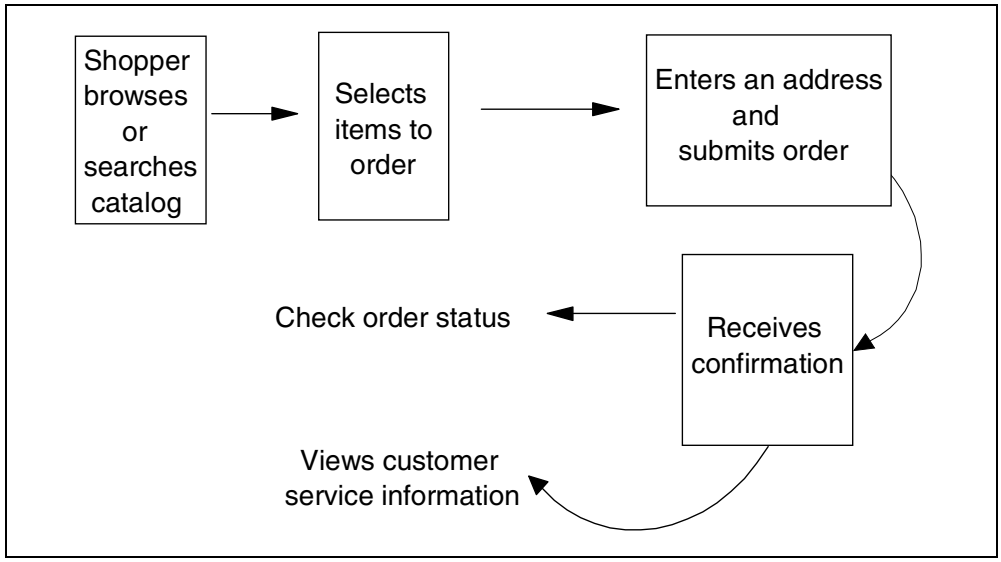


Figure 7. Electronic store shopping model

To enable this process, such functions as shopping cart, shipping, product catalog, order tracking, user registration, taxation, back office integration, order processing, credit cards, and payment must be integrated into the application. These are not standard features of Domino but can be built using the tools provided with Domino.

1.4.3 Customer readiness and expected added value

Customer readiness is very important for any e-business project. You need to review the following aspects:

- **Level of technical skills**: Is the customer I/T team able to support an e-business project on its own? Or can they rely on external solution or service providers (Business Partners) that can help them implement the new application and corresponding architecture?
- **Attitude towards new technology adoption**: Will new tools or a new user interface be easily accepted inside the organization (or outside, at least as far as Business Partners, distributors, and so on are concerned)?
- **Degree of risk acceptance**: The customer's confidence to move on new technology adoption will have an impact on the adopted architecture of the application, or on the time the application is actually started.

What will be the added value for the customer? How will this e-business solution:

- Increase the number of customers?
- Enable reaching new markets?
- Enhance the image of the company?
- Get products to market faster?
- Increase revenue or decrease costs?
- Enhance customer relationships?
- Improve customer service?
- Lower operating costs?
- Make employees more effective in teaming and collaboration?

These kind of considerations need to be discussed, and decisions need to be made, when starting to develop an e-business application.

1.4.4 Implementation examples

Many customers, as well as Lotus and IBM Business Partners, have used Domino for AS/400 to develop e-business applications.

Several implementation examples using Domino for AS/400, are presented in the *AS/400e e-business Handbook*, SG24-5694, in the “Lotus Domino for AS/400” chapter.

Section 8.4.4, “Scenarios: Positioning, sizing, performance, and implementation”, in the *AS/400e e-business Handbook* includes write-ups of selected case studies, that is, actual customers using Domino for various functions of e-business. The problem to solve and the solution chosen is described for each case study selected.

Chapter 2. Domino for AS/400 infrastructure

Lotus Domino is the world's leading workflow, messaging, groupware, and Web software. It enables you to communicate with colleagues, collaborate in teams, and coordinate strategic business processes on and off the Web.

Using the strengths of both AS/400 and Lotus Domino, you can enhance existing business applications and increase collaboration and coordination in your organization. Domino for AS/400, known for its reliability, scalability, and integrated security, fits well into the e-business application framework. It can be used to create, deploy, and manage e-business solutions.

2.1 Domino 'Internet' servers

The following Domino "Internet" servers are described in this section:

- Domino HTTP server
- Domino messaging servers
- Domino LDAP server

2.1.1 Domino HTTP server

Lotus Domino for AS/400 has an HTTP server that is a high-performance, Web server capable of serving static HTML pages, images, audio, video, and traditional Notes design elements, such as databases, views, forms, agents, and documents. It also has a servlet engine that enables it to serve up Java servlets.

The HTTP server runs as a task within the Domino application server. It can be stopped and started without having to restart the entire Domino application server.

As a Web application server, Domino can host Web sites that both Internet and intranet clients can access. It can also serve pages that are stored in either the file system or in a Domino database.

When a Web browser requests a page in a Domino database, Domino translates a document into HTML. When a Web browser requests a page in an HTML file, Domino reads the file directly from the file system. Then the Web server uses the HTTP protocol to transfer the information to the Web browser. Using Domino to store Web pages as documents in a database has a major advantage over storing static HTML pages. Using Domino, any change that you make to a database is automatically reflected on the Web server.

Any Domino application can be a Web application. Before you create a Web application, become familiar with the Domino features that can be translated into HTML and determine whether Web browser users, Notes clients, or both will access the application.

Domino includes these Web server features:

- Translation of Notes features into HTML code. For example, in HTML code, hotspot links are translated into anchor (<A>) tags.
- Passthru HTML. HTML code that you include in a form, document, or About and Using documents and that Domino interprets during the page translation.

Passthru HTML lets you use Web-only text formatting, links, images, commands, and programs. Using passthru HTML, you can combine Domino features with HTML code.

- Security for applications using standard Domino security, such as the database ACL, and Internet security features, such as Secure Sockets Layer (SSL) and name and password authentication.
- Support for Java applets that are either referenced using passthru HTML or embedded in a document.
- Support for JavaScript that is either included as passthru HTML or embedded directly in a document.
- Support for CGI programs that are referenced using passthru HTML in a document.
- Support for static HTML pages that are referenced in a directory on the server's hard drive. Static HTML pages can be referenced by passthru HTML included in a document or can be requested directly using a URL.
- Support for URL extensions that expose Domino functionality to the Web client, for example opening a database or view.
- Redirecting and remapping URLs and directories to another location.
- Support for virtual servers, which allow more than one Web site with a separate DNS name to exist on a single server machine.
- Support for server clusters, which allow a server to failover to an answering server if the first server is unavailable and provides load balancing to maximize response time for users.

The Domino Web server examines the URL in the incoming request. It determines if the request is for an item in a Domino database or if it is a request for an HTML file in the file system. If the request is for an HTML file, Domino acts just like any other Web server and serves the file to the Web client. When the request is for something in a Domino database, Domino interacts with the database to serve the information to the Web client or to put information from the Web client into the database.

Domino supports URL extensions that expose functionality to the Web client. For example, this URL opens the main database on the Notes.net site:

```
http://www.notes.net/welcome.nsf?OpenDatabase
```

Note

The appendix “Programming a Web site” in *Domino Designer Release 5*, Part Number CT6E2NA, contains information about Domino URL commands. Other programming tools, such as formulas, Java, JavaScript, Perl, and CGI scripting are also described in this appendix.

2.1.1.1 Internet Cluster Manager

To increase the availability of the Domino HTTP server, Domino R5.0 Enterprise Server includes the Internet Cluster Manager (ICM), which extends the failover and load balancing capabilities to Web browsers (for example, Microsoft Internet Explorer and Netscape Communicator). As shown in Figure 8, you can cluster

together multiple Domino servers and provide failover and dynamic load balancing for both Notes and Web clients.

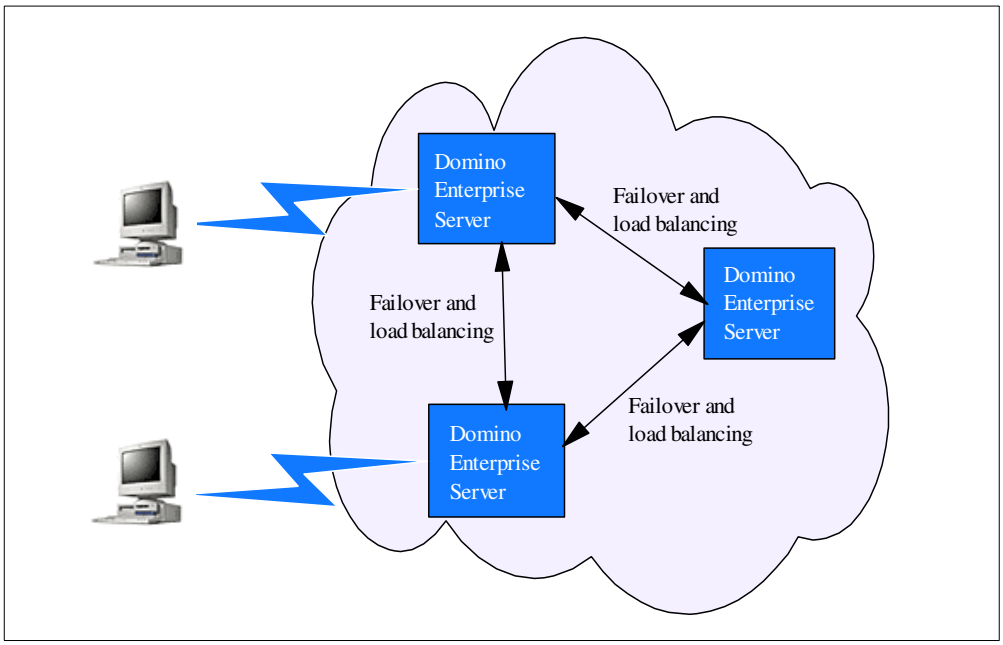


Figure 8. Domino Internet clustering

Domino clustering provides “content-based routing”. This means Domino performs intelligent failover and load balancing based on the content requested by the user. To do this, Domino uses event-driven Cluster Replication to ensure that all replicas of databases are always kept in sync. Then, the Domino Cluster Manager maintains information on the replicas and on the server availability to route users to the best server.

Internet Cluster Manager (ICM) is a new Domino R5.0 server task that's behind the failover and load balancing capabilities. The ICM serves as a liaison between the HTTP clients and the HTTP servers of a Domino cluster. The HTTP clients direct requests for a database to the ICM. The ICM maintains information on the availability of the Domino servers in the Domino cluster, and also maintains information about the distribution of databases on the servers. The ICM determines the best server to receive a particular client request and directs the request to that server.

The ICM can:

- Monitor backend Domino servers for availability
- Check the Domino HTTP Web service for availability
- Disallow any new connections to servers out of service
- Provide failover (direct clients) to the best available server
- Provide load balancing by setting availability thresholds for your Domino servers
- Support “virtual IP addresses” and map ports
- Provide content routing for your clients

Why use the ICM in addition to Domino clustering? There are two main reasons. Your organization may want to provide high availability to its customers using browsers to access Domino Web applications. Therefore, you would install the ICM to provide failover support for your Domino servers. Your organization may also want to provide scalability for its Domino Web applications. Therefore, you would install the ICM and use load balancing of browser clients accessing your Domino Web applications.

2.1.1.2 Planning for your ICM installation

The characteristics of the ICM are end-to-end security and authentication. If you choose to require a name and password, and a SSL connection on your servers, the ICM preserves SSL encryption and decryption. The ICM supports the current Domino security model used for Web access. The HTTP requests are sent with no user ID or password to the ICM. The ICM then redirects requests at the protocol level and sends the redirection response code and target server information directly back to the HTTP client. The HTTP client then issues a request to the specified target Domino server, and the local Domino security controls take effect. This may result in authentication dialogs between the HTTP client and the target Domino server.

If the user is successfully authenticated, the user ID and password are supplied in the HTTP headers for all subsequent requests to that Domino server. The ICM is not involved in the user identification and authentication dialog.

The ICM uses SSL for transport-level security. You can configure the ICM to require it to use SSL sessions. The ICM may use the same certificates the browser and Domino HTTP server use. It is designed not to introduce new administrator tasks for distribution and maintenance of these certificates, beyond what is required to allow a Web client to connect to a Domino HTTP server using SSL. The ICM is not a firewall. Therefore, if it is necessary to protect the ICM from unauthorized access, you may choose to additionally install a firewall.

The ICM does not require any additional hardware. You can install it on your Domino server, and it runs on all OS and hardware platforms supported by the Domino server. The ICM supports client connections through TCP/IP only. You can use any supported Notes protocol between the ICM and the servers in a cluster.

The ICM is responsible for managing HTTP and HTTPS requests to the Domino server. It does not manage FTP, SMTP, or UDP. There are other load balancing hardware or software products that manage these protocols.

2.1.1.3 ICM failover and load balancing from the browser perspective

The ICM provides simple, but strong affinity of the browser client with the server. Since the ICM uses redirection, the browser directly connects to the server that hosts the information requested. Since the ICM redirects the connection to the server, the client is directly connected to the server during the course of a user's session that involves information on that server. Multiple requests are made to that server, maintaining the client's state information.

The browser client may experience failover when the Domino server is unreachable due to hardware, software, or network failures. Failover can also occur when the Domino server is restricted or busy, or when the database (or replica) is unavailable because it is either marked “out of service” or “pending

delete". When the browser client is redirected to a server and that server subsequently fails, the client times out and generates a "server not responding" message to the user. The actual text of the message depends on the client. The browser then returns to the ICM to be redirected to an available server. This can happen in a variety of ways, such as clicking the Back button of the browser or simply retyping the URL. The user may be required to authenticate with the new server. If the user previously authenticated to the new server during the browser session, perhaps to access another database, no re-authentication is needed. If both the browser and server support SSL3, and the user selects this option, re-authentication occurs automatically.

URLs currently supported by the Domino HTTP server that are also supported and processed by the ICM include:

- Open servers, databases, and views
- Open forms, navigators, and agents
- Open, edit, and delete documents
- Open documents by name from a view
- Open image files, attachments, and OLE objects
- Create search queries

When an HTTP client makes a request to the ICM that is not targeted at a Domino database, the ICM selects an available server in the cluster that is running a Domino HTTP server and redirects the HTTP client to that server. Files not stored in a Domino database should replicate across all servers in the cluster.

Domino HTML generation now directs HTTP clients to the ICM for databases within a cluster, if that cluster is configured with ICM support. The ICM knows the location of the databases because the ICM has access to the Cluster Database Directory (CLDBDIR.NSF). This directory hosts information about all the databases in the cluster and all the servers on which they reside, including database availability indicators. Failover from a browser perspective can be seen in Figure 9.

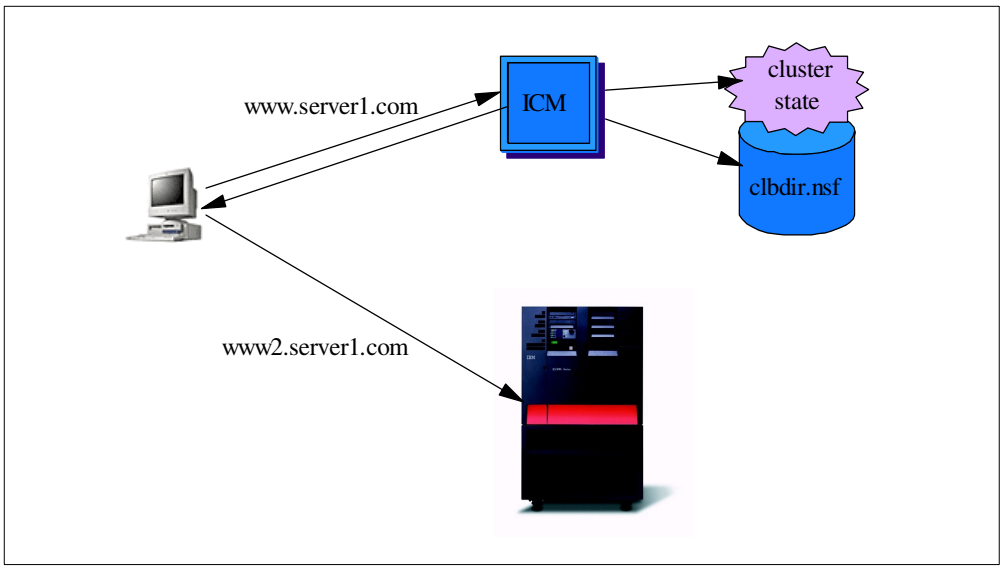


Figure 9. Internet clustering from a client perspective

2.1.2 Domino Internet messaging servers

Domino R5.0 allows all types of clients to enjoy full-fidelity Internet messaging. With support for native MIME content and native SMTP routing, Domino integrates the features that previously required a separate message transfer agent (MTA). The native support means that conversions between the Notes format and the Internet format are no longer required. In addition, the server includes a new directory catalog for faster name lookups and expanded LDAP support for more directory features.

To allow messages to be more than just plain text, the Multipurpose Internet Mail Extensions (MIME) standard “extends” the format of messages to allow for audio, video, international character sets, and multi-part messages. Most Internet messages are stored in the MIME format. In Domino R5.0, all Notes messages are stored as either Notes Rich Text or MIME. Domino translates dynamically between the formats as needed. This means that Notes users can send mail to and from Internet users with no loss of fidelity. In addition, Domino supports S/MIMEv2 for secure Internet mail delivery, including encryption and digital signatures.

To transfer mail reliably and efficiently, the Simple Mail Transfer Protocol (SMTP) defines mechanisms for relaying mail between networks on the Internet. Previously, you could configure the SMTP MTA on a few dedicated Domino servers to act as gateways for your Internet mail, converting Notes mail to Internet mail, and vice versa. All Internet mail needed to transfer through these servers. With Domino R5.0, you no longer need to install a separate MTA. All Domino servers now have native SMTP capabilities, so all servers can transfer mail directly to and from the Internet. Also, as mentioned previously, conversions to and from MIME are no longer necessary since the R5.0 client can send and receive native MIME messages. The Domino server can convert messages for pre-R5.0 clients without any problems. Domino also supports Extended SMTP (ESMTP) for delivery notifications for Internet messages, for example, reporting whether deliveries are successful, failed, or are delayed. In addition, Domino R5.0 allows you to secure SMTP connections using TCP/IP or a TCP/IP port secured with Secure Sockets Layer (SSL). You can require a name and password either unencrypted over TCP/IP or encrypted over SSL.

Domino R5.0 also supports native Internet addressing, based on the Internet RFC821/822 standards for mail addresses. RFC821 specifies an address of localpart@domain, while RFC822 adds a “phrase” portion to the address that looks like “phrase” <local part@domain>. In Domino R5.0, the phrase portion of the address contains the user's distinguished name for backward compatibility (for example, "John Doe/Acme" <jdoe@acme.com>).

Domino R5.0 includes the following features in the mail router:

- Controls who can route documents over a given size and when
- Configures the router to obey database quotas
- Uses anti-spam controls to prevent unwanted mail from being forwarded through your domain

- Allow users to connect and “pull” mail using temporary network connections, for example, when dialing into an Internet Service Provider (ISP)
- Enjoy a greater amount of concurrency, with support for multiple sessions to the same destination, multiple delivery threads, and synchronous mail delivery agents

Domino R5.0 has a POP3 server and an IMAP server to facilitate the accessing of mail boxes using either of these protocols.

2.1.3 Domino LDAP server

Domino R5.0 expands its support of the Lightweight Directory Access Protocol (LDAP), which makes the Domino Directory available for searching over TCP/IP. Domino supports LDAPv3 with authenticated read/write access. This means that, for example, you can use the LDAP write operations to make changes to the Domino Directory through an LDAP client. In addition, the Domino Directory scales to support at least one million registered users in a single domain.

For authentication, you can use LDAP to authenticate users in external directories, and to authenticate members of a group. For example, you can use an external directory to check passwords and X.509 certificates for your Web applications. You simply define a trust relationship in Domino's Directory Assistance database (previously called the Master Address Book). Then users can use the LDAP-based external directory defined in Directory Assistance for authentication and access control information.

Domino R5.0 supports the LDAP Data Interchange Format (LDIF) for importing and exporting directory information between LDAP servers. An LDIF file contains a series of records, each of which describes a directory entry. After you export a directory to a file, you can then use the file with an import utility to import its contents into another directory. For example, you can import directory entries from an LDIF file into the Domino Directory to create new user accounts during user registration.

You can also export a directory to see its schema. This means that you can see the schema of the Domino Directory and modify it, for example, to add fields required in the directory by other applications. Any changes that you make to the directory are now automatically preserved, as long as you follow the rules for schema extension outlined in the Administration Help.

More information about Domino directory infrastructure is available in Chapter 3, “Managing directories and authentication” on page 57.

2.2 Integration with AS/400 applications

In most organizations, data and information are stored in a number of systems, like Relational Database Management System (RDBMS), Customer Relationship Management Systems, and others. It’s important to easily access data from all these systems and to build business applications based on that data. In view of this, Lotus has developed a number of tools that can be used with Domino to access these systems and their data. Data and information outside Domino can now be easily accessed and integrated into Domino applications.

There are a number of services for connecting Domino R5.0 to enterprise data and information. Some of these services include:

- Domino Enterprise Connection Services (DECS)
- Lotus Enterprise Integrator (LEI)
- Lotus Connector LotusScript Extension (LSX LC)
- Lotus Connector Java classes

Lotus Enterprise Integrator (formerly NotesPump) is an optional tool running on top of the Domino server, which is available for exchanging data between data sources either on schedule or on demand. A variety of data sources and activity types is supported.

DECS, which is a function provided within Domino for AS/400, is functionally similar to the RealTime Notes activity in LEI. DECS does not replace LEI.

For LotusScript and Java programmers, the LSX LC and LC Java classes provide a common interface to all enterprise systems. You can still use LotusScript:Data Object (LS:DO) and @DB functions (both described in 5.4, “Tools for enterprise integration” on page 117), MQ Series, and SAP R/3 LSXs.

Underlying these services are individual connectors for the supported enterprise systems. Figure 10 shows Lotus Domino enterprise integration architecture. It shows where the different services fit in to the architecture.

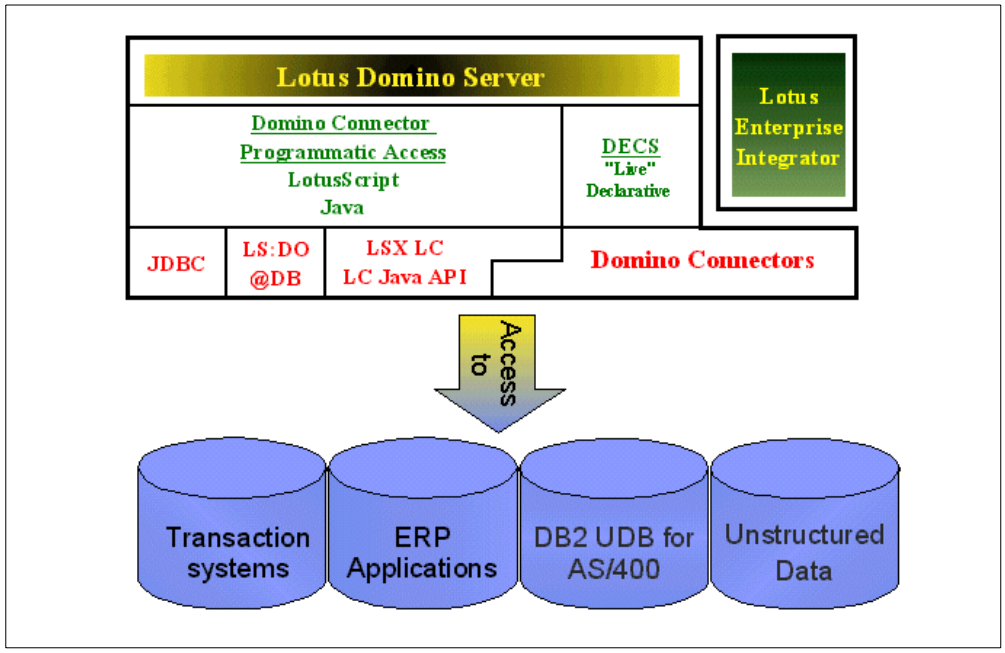


Figure 10. Domino for AS/400 enterprise integration architecture

2.2.1 Lotus Domino Connectors

Lotus Domino Connectors exist to permit access to external data sources from Domino for relational database management systems, enterprise resource planning systems, transaction processing systems, directory services, and other services. Connectors are components that provide a consistent interface to a variety of systems. Some standard connectors are included with the Domino server, and add-on connectors can be purchased separately from Lotus or

third-party vendors. Among the standard connectors are those for Notes, DB2, and files.

Some of the available, or soon to be available, Enterprise Resource Planning (ERP) connectors for the AS/400 system are:

- Lawson Enterprise/400 integrates Domino with Lawson Enterprise/400 systems.
- SAP R/3 integrates Domino and SAP/R3 applications. Reading and writing data is always through the application layer, maintaining all RFC and R/3 Transaction business rules.
- J.D. Edwards OneWorld integrates Domino with J.D. Edwards OneWorld ERP systems. The connector performs data transfers through the application layer to preserve all business logic validations. The Domino or LEI machine must contain J.D. Edwards OneWorld Client Version B732 or later.
- SSA is also planning to develop a connector for BPCS on the AS/400 platform.

2.2.1.1 LotusScript Extension for Lotus Domino Connectors

The LotusScript Extension for Lotus Domino Connectors (LSX LC) provides programmatic access to Lotus Connector enterprise data through a common set of classes. The programmer uses a single model no matter the enterprise source. Programming provides more control and additional capabilities over DECS and LEI. The classes in the LSX LC use the same connectors as DECS and LEI to access the enterprise data.

LSX LC is discussed in more detail in 5.4.3, “LotusScript extension for Lotus Domino Connectors” on page 122.

A Lotus Connector Java API is also available. It is delivered in Lotus Enterprise Integrator.

2.2.2 Domino Enterprise Connection Services

Domino Enterprise Connection Services (DECS) extends the Domino application to query and modify enterprise data. When a Notes or Internet client accesses data through a Domino form connected to an enterprise system, DECS transparently retrieves and updates the enterprise data. The client needs no additional software and needs no knowledge of the enterprise system. DECS requires no programming.

DECS is discussed in more detail in 5.4.5, “DECS” on page 125.

2.2.3 Lotus Enterprise Integrator

Lotus Enterprise Integrator (LEI), which is an add-on product for Domino for AS/400, performs data transfer, data synchronization, and other activities between data sources. A data source can be Domino or any supported relational database or enterprise system. Data activities occur on a scheduled or event-driven basis and are capable of high-volume, high-speed transfers.

LEI is discussed in more detail in 5.4.6, “Lotus Enterprise Integrator (LEI)” on page 127.

2.2.4 Java and Domino R5.0

Domino for AS/400 contains various Java functions for enterprise integration. Some of these are JDBC, servlets, and CORBA. The AS/400 Toolbox for Java can also be used.

Some Domino for AS/400 Java integration options are discussed in more detail in 5.4.4, “Java integration options” on page 124.

2.2.5 Considering Domino for AS/400 on a frontend server

Using a Dedicated Server for Domino, or another AS/400 system, as a frontend to an AS/400 system running Line of Business applications and DB2, is a solution that should be considered in some situations. You should consider it even if the response time isn’t as good as the response times obtained on a single machine.

From a programming perspective, you can use one or more of the normal Domino techniques for accessing relational data: @DB functions, LS:DO, LSX LC, JDBC, or DECS. Lotus Enterprise Integrator allows scheduled, event-driven, and bi-directional exchanges of data between IBM DB2 UDB for AS/400 and Domino for AS/400. Under the covers, these techniques use AS/400 Distributed Relational Database Architecture (DRDA) support when the relational data resides on another server (Figure 11).

The redbook *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345, provides information on how to setup a DRDA connection over TCP/IP or SNA (refer to the “DRDA Connection Setup” appendix) and examples of using DRDA with LEI and NotesPump (refer to the “Using DRDA with LEI and NotesPump” appendix).

The white paper “Evaluating Appropriate Workloads for the AS/400e Dedicated Server for Domino” discusses performance aspects. It is posted on the Web at:
`http://www.as400.ibm.com/domino`

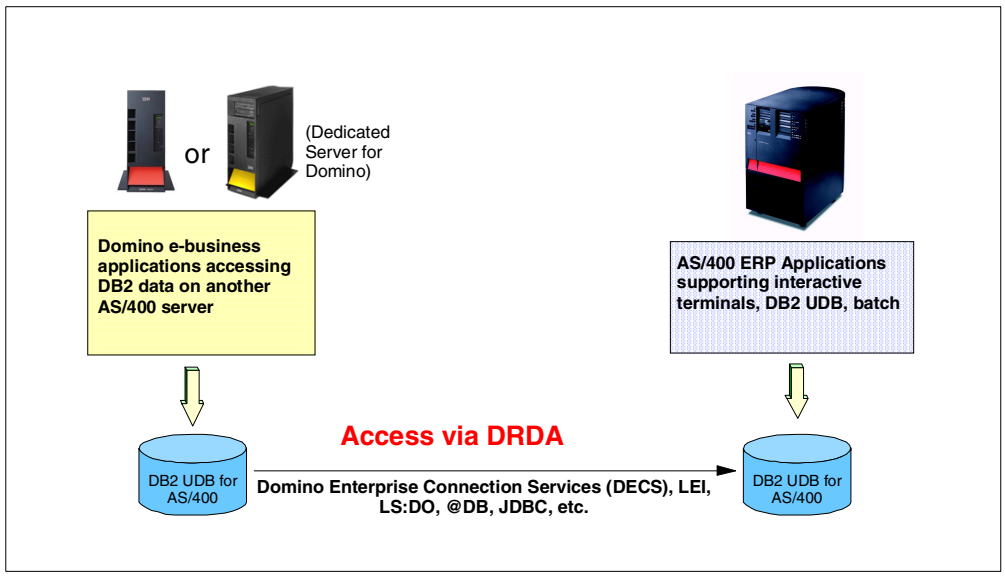


Figure 11. Domino for AS/400 on a frontend server

2.3 Network

Just as important it is to choose the right server when it comes to e-business, it is important to get the right network implementation to support your e-business environment. If you don't get the network right, you could pay the penalty of having bad performance. A network typically consist of:

- Servers
- Clients
- Hubs
- Routers or gateways
- Firewalls

There are typically two scenarios that correspond to the type of project we are explaining in this redbook:

- Running Domino on a separate AS/400 system than the AS/400 system running DB2 UDB for AS/400 (shown in Figure 12)
- Running Domino and DB2 UDB for AS/400 on the same AS/400 system

Between the two scenarios, there are different server setup requirements, for example:

- How it must be sized
- How to handle and adjust performance
- How to secure the servers and the network

We will look at the server element, since it is out of this redbook's scope to cover all of these elements.

In this section, we concentrate on what we have to consider when choosing the right AS/400 server, and what we can do to improve the performance of this AS/400 system being on the network. Knowing how to configure TCP/IP and Domino on the AS/400 system can benefit the overall performance of the network.

Guidelines for how to configure your hubs, routers, gateways, and firewalls to optimize performance should be provided in the documentation you have for the different products. Some of the hints given here can be transferred to these other elements.

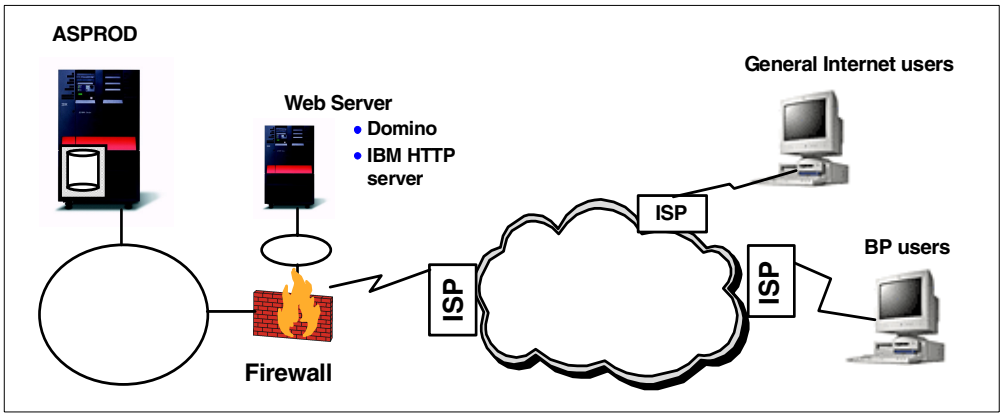


Figure 12. The network

2.3.1 Performance considerations

There are many factors that can impact overall performance (for example, end-user response time, throughput) in the AS/400 Domino environment, some of which include:

- AS/400 processor speed
- Utilization of key AS/400 resources (CPU, IOP, memory, disk)
- Object contention (for example, mutex waits, lock waits)
- Speed of the communications links
- Congestion of network resources
- Processing speed of the client system

The primary focus of this section is to discuss the performance characteristics of the AS/400 system as a server in a Domino environment, providing recommendations for best performance. These recommendations are divided into two areas:

- AS/400 Domino general performance
- AS/400 Domino Web serving performance

These areas also covers how to set up the AS/400 system in regard to network performance.

2.3.1.1 AS/400 Domino general performance

As you develop your Domino applications on the AS/400 system, there are some simple steps you can take to ensure you are receiving the best performance from your AS/400 Domino server.

The tuning tips given are listed in the order of potential gain. As we progress down the list of tuning tips, the amount of performance gain will decrease. It is important to implement all of the tuning recommendations to fully enable your AS/400 Domino server to perform optimally.

Multiple Domino partitions on one AS/400 system

One performance benefit of having multiple partitions is the ability to split different types of Domino users into their own partition/server. For example, you can have your mail users in a Domino subsystem, so they are not competing for the same resources of users in another Domino subsystem that is doing another Notes function such as Domino database accesses, including heavy text search, for example.

By splitting your Notes users into different subsystems, you also increase the availability of your AS/400 system. If one of your Domino servers goes down, only that subsystem is affected. All of your other subsystems stay up and running.

A tuning recommendation is to limit the number of active users to 1,000 to 2,000 per Domino partition. This limits the number of total active threads and therefore optimize the performance of your Domino server by reducing contention among the open threads.

NOTES.INI settings

There are a number of variables that we recommend setting in the NOTES.INI file. Unless you explicitly add these variables to your NOTES.INI file, the default values for each of these variables will be used. Your Domino server needs to be restarted after these entries are added to the NOTES.INI file to pick up the

changes. The variables that need to be changed and the recommended value to set them at are shown in the following list. These are the NOTES.INI variables that are recommended to change for all Domino subsystems.

- **SERVERTASKS**: The SERVERTASKS variable controls which tasks are started when a Domino server is brought up. By limiting the number of tasks that start automatically with the Domino server, or checking that the Domino server does not load tasks that are not used (for example billing), you can improve the performance of the server. If you have partitioned Domino servers to handle different Notes functions, only start the tasks that are needed in each partition.

  Note

  It is very important that you *do not* to remove the UPDATE task. If you remove the UPDATE task, you can have functional problems in your Domino server.

- **NSF_BUFFER_POOL_SIZE**: The NSF buffer pool contains cached information on the server. By changing the value of this variable, you can control the size of the memory section used for buffering input and output to disk storage. It is very important to set this variable if your Domino applications use indexing. There are two ways to set this variable, NSF_Buffer_Pool_Size and NSF_Buffer_Pool_Size_MB. If using NSF_Buffer_Pool_Size, you need to specify the correct number of zeroes, such as 250000000. It is certainly easier to use NSF_Buffer_Pool_Size_MB since the same setting would be specified at 250. The recommended setting varies depending on which release of Domino you are running on the AS/400 system and the number of partitioned Domino servers that are being used.

  For Domino Release 5.0, we recommend that you set NSF_Buffer_Pool_Size_MB to a value of 25% of the pool size that the Domino subsystem is running in, or a maximum of 300 MB.

  What if you have multiple Domino servers running in the same memory pool? The general rule here is that you want to give each Domino server's NSF_Buffer_Pool_Size_MB as much memory as possible, but you do not want to over-allocate the memory that is available in a memory pool. The recommended setting is to look at the total amount of memory that is available in the memory pool, take 75% of that total memory and that is what you have available to distribute among the Domino servers. Divide that amount by the number of Domino servers you have in that memory pool.

  You can use the `SHOW STAT` command on the Domino console to obtain the following information:

  – Buffer pool maximum
  – Buffer pool used
  – Buffer pool peak

If the peak is less than 95% of the maximum, the index is performing optimally within the buffer. Use the reported tasks to capture statistics per server over a period of time, and graph them perhaps to identify trends and peaks.

2.3.1.2 Editing the NOTES.INI file

The NOTES.INI file contains many settings that Domino relies on to work properly. An accidental or incorrect change may cause Domino or Notes to run unpredictably. Therefore, you should edit NOTES.INI only if special circumstances occur, or if Lotus Customer Support recommends that you do so.

There are several ways to edit NOTES.INI settings:

- Open the NOTES.INI file and edit it. The procedure for doing this depends on your client's or server's operating system and the text editor you use.
- Create a Configuration Settings document, and edit its settings. Using a Configuration Settings document, you can add and modify many NOTES.INI settings at a time. However, there are a number of settings that you cannot set in the Configuration Settings document. Also, because a Configuration Settings document applies only to Domino servers, you cannot use it to modify a Notes client's NOTES.INI file.

Because directly editing the NOTES.INI file is unsafe, we recommend that you save it before any change. Lotus recommends that you use the Configuration Settings document to modify server settings.

On the following pages, we show you a set of displays that demonstrates how to edit your NOTES.INI file and set these variables to the recommended values. The placement of these variables in the NOTES.INI file can be anywhere you want to put them. It may be easiest if you group them near the end of the file for ease of finding them in the future. We recommend that you add them prior to the last entry in the NOTES.INI file because sometimes the last entry is not recognized. This is due to how the entry was made, whether the person who adds these entries presses Enter or the field exit key after the last entry.

Following is a description of editing the NOTES.INI file from AS/400 Operations Navigator.

Note

To edit the NOTES.INI file from a 5250 display, enter the Work with Domino Servers (`WRKDOMSVR`) command. Then select the server and then enter option `13`, Edit NOTES.INI, in front of the selected server.

For details, see the Domino for AS/400 Release Notes, provided as a Domino database, in the document *AS/400 PTF requirements*.

The Domino for AS/400 Release Notes (readas4.nsf) database contains last-minute information specific to the AS/400 platform. A printed version comes with the Domino for AS/400 CD-ROM. The Web site at `http://notes.net/doc` may have an updated version.

Figure 13 shows where to find the Domino server properties in AS/400 Operations Navigator. A plug-in is required to manage Domino from AS/400 Operations Navigator. How to install the necessary plug-in for doing this is covered in the “Graphical user interface of Operations Navigator” chapter in *Lotus Domino for AS/400 R5: Implementation*, SG24-5592.

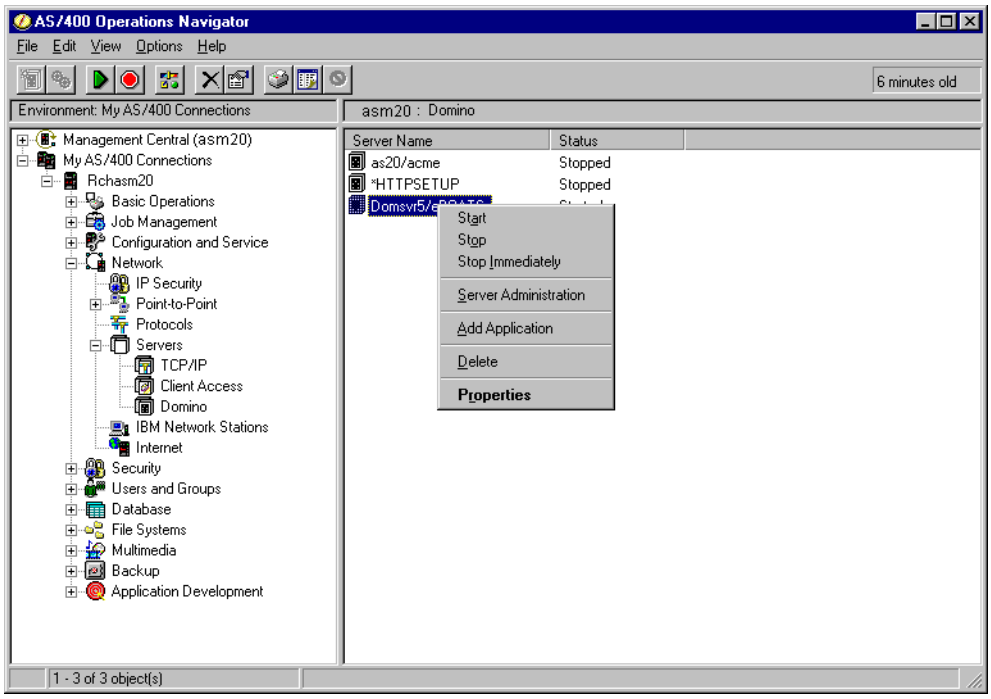


Figure 13. Operations Navigator support for Lotus Domino

Click **Properties**, and the window as shown in Figure 14 appears.

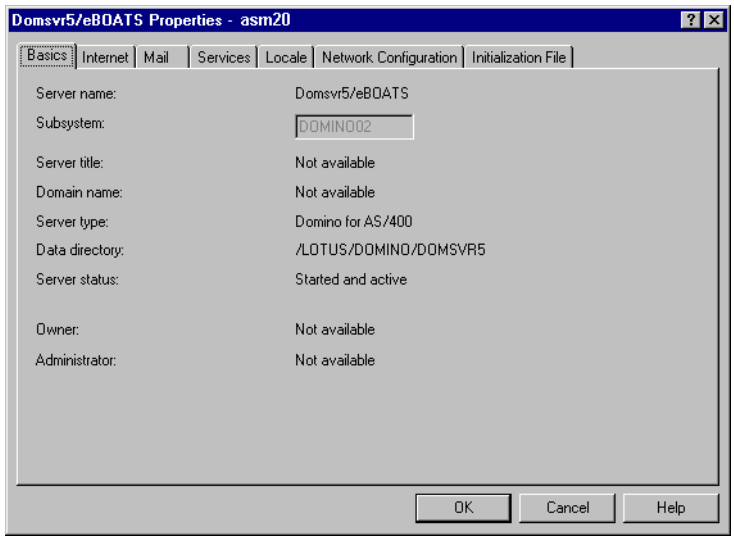


Figure 14. Domino server name properties

Click the **Initialization File** flag. This shows you the NOTES.INI file (Figure 15 on page 40).

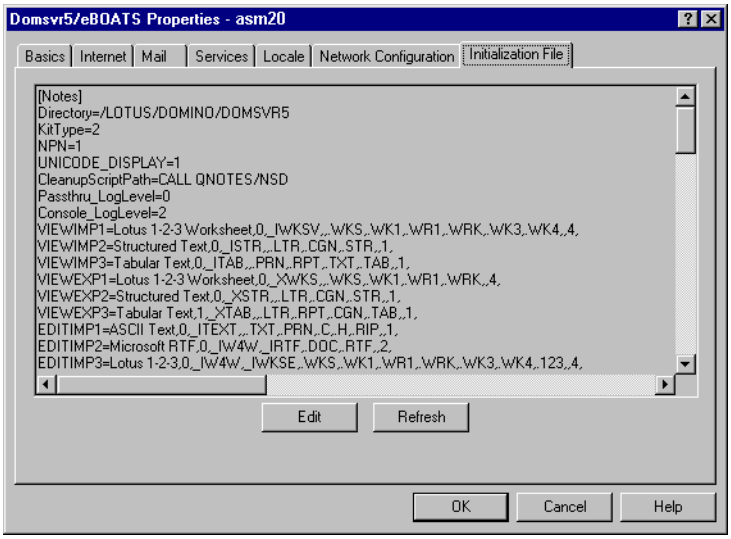


Figure 15. The NOTES.INI file

To edit the file, click the **Edit** button. The message shown in Figure 16 appears.

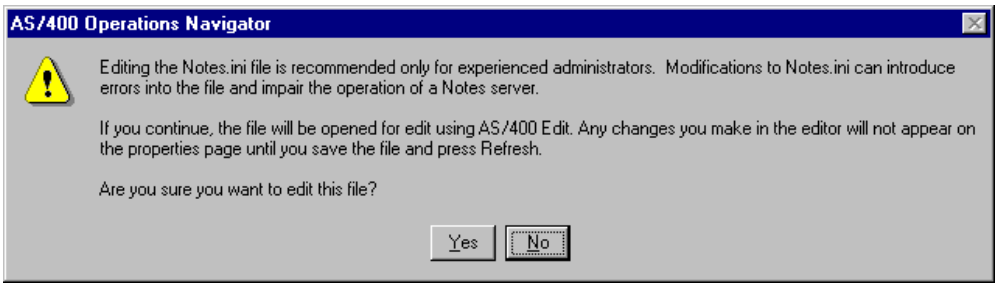


Figure 16. Edit NOTES.INI warning message

To continue, click the **Yes** button. Then, you are prompted for a user ID and password before you access the editor for the NOTES.INI file as shown in Figure 17.

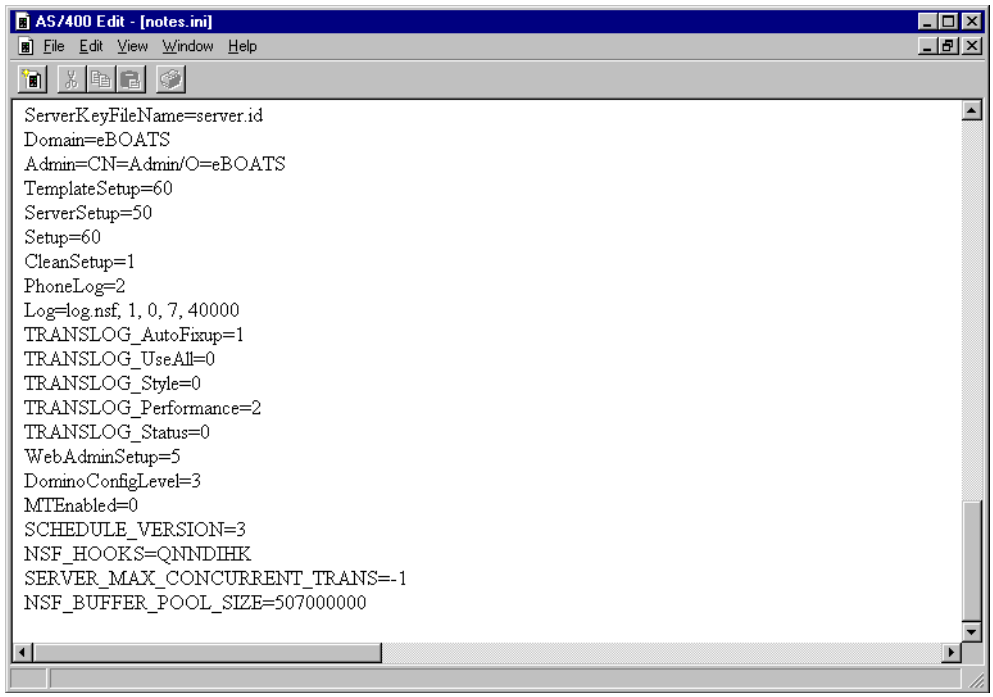


Figure 17. Editing the NOTES.INI file

Once the file is open, the first variable mentioned for tuning is the SERVERTASKS variable. Here you can add and remove server tasks from automatic start up. Such server tasks as POP3, LDAP, IMAP, Billing, Replica, or even AMgr may not need to run in your Domino server.

You also add the variable NSF_BUFFER_POOL_SIZE.

Save and Exit the NOTES.INI file in the editor. Then, you return to the next window, which is shown in Figure 18 on page 42. Click the **Refresh** button to see that the NOTES.INI file has been updated.

Don't forget to stop and restart your Domino server, so that any change to the NOTES.INI file is taken into account.

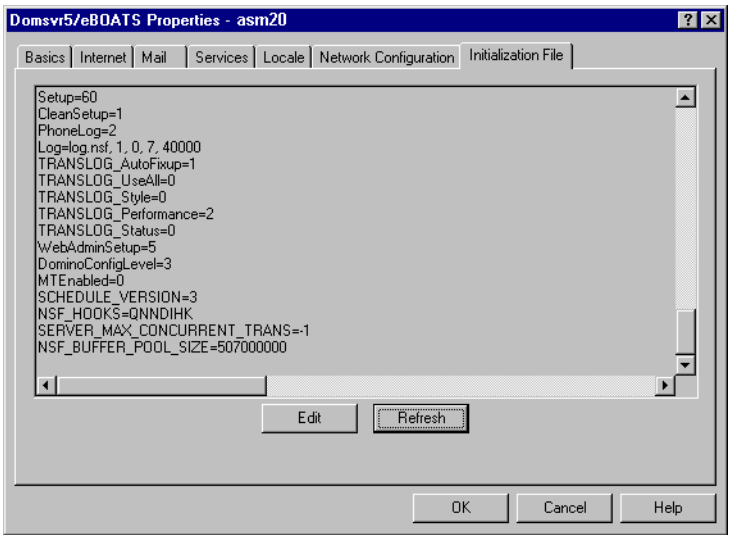


Figure 18. The NOTES.INI file after editing and refreshing

2.3.1.3 AS/400 Domino Web serving performance

There are many factors that can impact the overall performance seen by the end-user in any Internet environment. The response time that a user sees depends on the Web browser they are using, the communications network, and the Web server. This section helps to shed some light on these areas and provide guidance on how to obtain the best performance from your AS/400 Domino HTTP server.

This section has two parts:

- The first part focuses on how to best tune your Domino server for HTTP Web serving.
- The second part focuses on AS/400 communication specific tuning.

Note

The scope of this section is on server tuning tips. There are a number of application design changes that can have a significant impact on the performance of a Domino Web serving application. See `http://www.notes.net` for application design performance information.

Domino Web serving performance

There are some configuration items that you can change for your Domino server to improve its Web serving performance. We highlight them in the following list. All of these items can be changed by editing the HTTP Server section of the server's name and address for a particular Domino server. In Lotus Domino R5.0, you can properly access these sections through the Domino Administration client.

- **Basic settings section**: In the basic settings section of the HTTP server settings, in the Domino server document (Figure 19), there are three different

variables that can be changed to achieve maximum Web serving performance. These variables are shown in Table 7.

Table 7. HTTP basic setting variables in the Domino server document

Variable	Recommendation
Number active threads	This is the limit of the number of active threads that are available for browser users. When the limit is reached, the next browser request then has to wait for a free thread. The default setting is 40. We recommend that you change it to the maximum number of concurrent users that is anticipated. To determine what value to set this to, use the `show statistic` Domino command on the Domino console and look for Domino.ThreadsActivePeak. You need the value of Number Active Threads to be a larger value than the value of Domino.ThreadsActivePeak. A recommended maximum for Number active threads is 200 HTTP threads per Domino server. See Figure 19 for an example.
DNS lookup	DNS lookup, when enabled, shows the host names of clients rather than the IP address in the Domino log files and the log filter. We recommend that this variable remain at its default setting of *disabled*. By leaving this function disabled, you will save CPU resources on the server. See Figure 19 for an example.

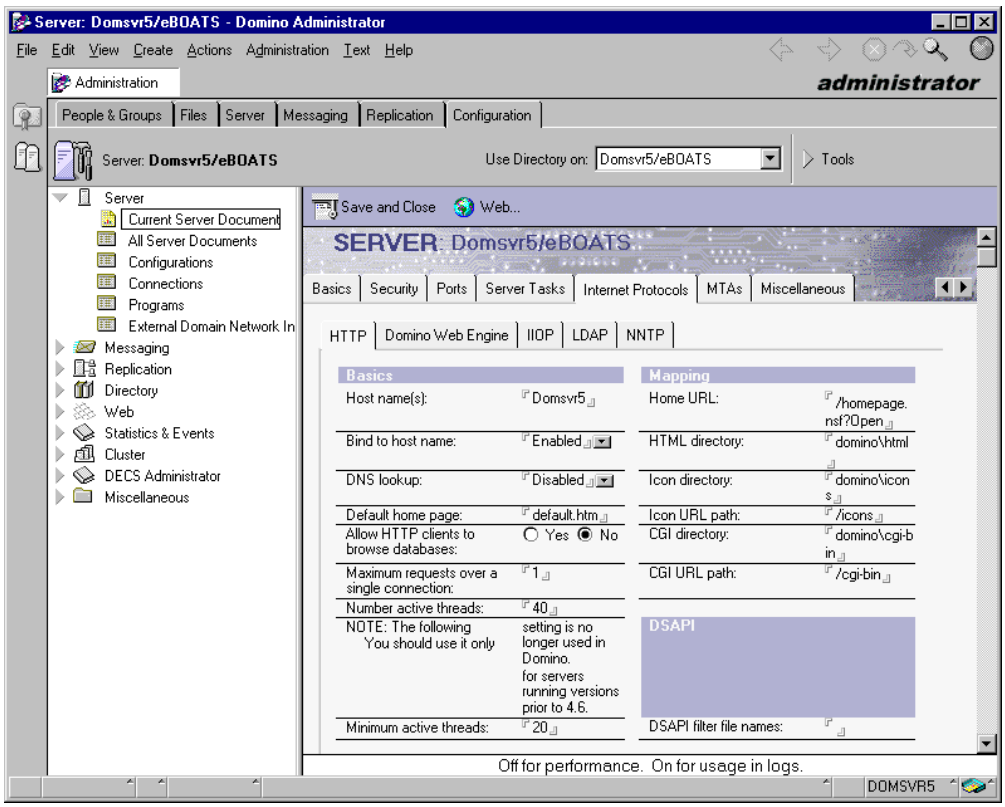


Figure 19. HTTP server basics settings

- **Logging parameters section**: The Enable logging *to* variables in the HTTP folder of the server document (Figure 20) are explained in Table 8.

Table 8. Enable logging to variables

Variable	Recommendation
Log files Domlog.nsf	The default for both of these variables is disabled. We recommend that you leave both of these variables at their default setting. Logging definitely has its purposes. However, it consumes extra CPU resources when it is enabled. For this reason, we recommend that these remain disabled for the best Web serving performance. The logging variables are found in the HTTP section of the Domino server document, as shown in Figure 20.

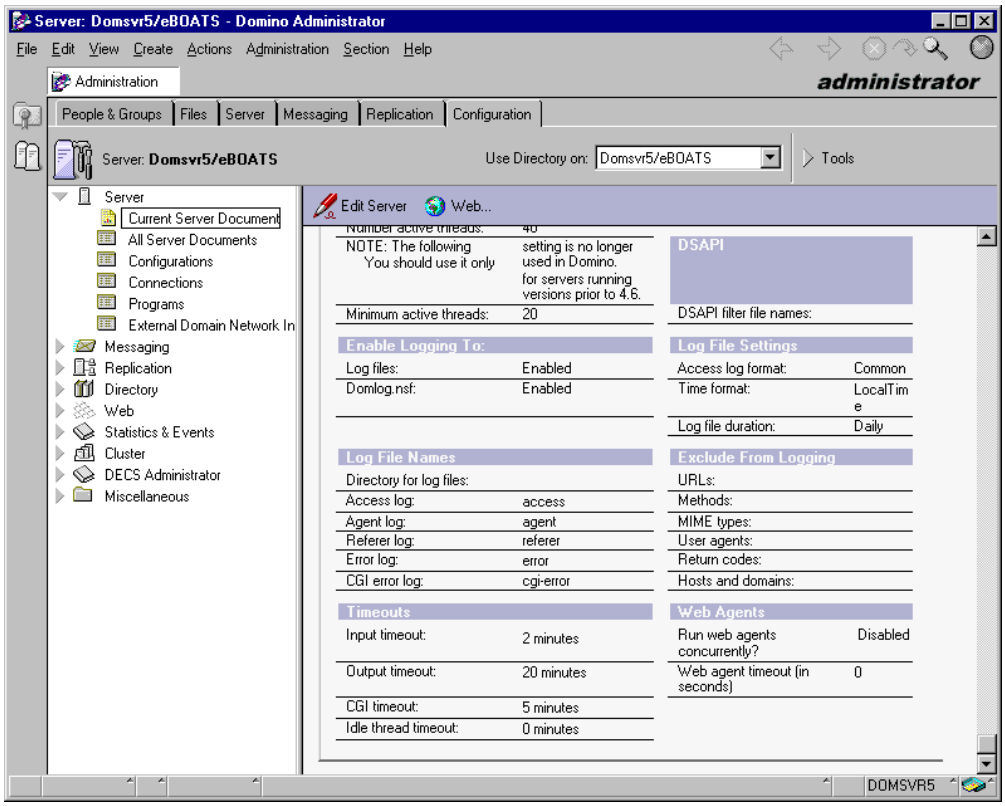


Figure 20. Logging parameters

When you are finished making the necessary changes to the HTTP server section and the Domino Web Engine section in the server document, remember to save it. For these changes to take effect, you have to restart your Domino server.

AS/400 network performance

Some of the communication tuning parameters that can be used on the AS/400 system to ensure you get the best Web serving performance from your system are explained in Table 9.

Table 9. AS/400 network parameters

Parameter	Recommendation
Maximum transmission unit (MTU) size	The maximum transmission unit size parameter affects the actual size of the line flows. We recommend setting this value to `*LIND`. which sets the MTU to the largest amount of data that can be transmitted on the line based on the line description information. By increasing the value of this parameter, you can reduce the overall number of transmissions, and therefore, increase the potential capacity of the CPU and the IOP. Use the `CFGTCP` command to change the MTU size. This dramatically improves Domino Web Serving performance since the HTTP client must pass data and the design and next response for every page that is served. Similar parameters also exist on the Web browser. The negotiated value will be the minimum of the server and browser (and perhaps any bridges or routers). Therefore, increase them all. To change the MTU size, enter the command `CFGTCP` on the AS/400 command line and press enter. At the Configure TCP/IP screen, select option **1** (Work with TCP/IP interfaces). From this screen, select option **2** (Change) for the interface that your Domino server is using. This takes you to the screen shown in Figure 21 on page 46. Make sure the Maximum Transmission Unit is set to `*LIND`.
Maximum frame (MAXFRAME) size	Maximum frame size is a parameter on the `CHGLINTRN` command that controls the size of the frame that is sent and received. The size of this frame can have quite an impact on the performance of the AS/400 controller. We recommend that you change this value to the maximum for the line description used for your Domino Web server. Change this variable to 16393 if are using a Token-Ring. To change the Maximum Frame Size, issue the `CFGTCP` command. At the Configure TCP/IP screen, select option **1** (Work with TCP/IP interfaces). This brings you to the Work with TCP/IP Interfaces screen. You need to write down the Line Description of the IP address that is being used by your Domino server. When you know the name of the Line Description, issue the following command and press F4: `CHGLINTRN LIND (Line_Description_name)` The screen in Figure 22 on page 47 appears. Change the Maximum frame size to 16393. The equivalent process must be done if you are using an Ethernet line.

Parameter	Recommendation
TCP/IP Buffer size	Web serving performance can be increased by increasing the buffer size that is used by TCP/IP, especially when sending larger amounts of data. The TCP/IP buffer size can be adjusted by using either the CHGTCPA command the CFGTCP command. Then, select option **3** (Change TCP/IP attributes) and change the TCPRCVBUF and TCPSNDBUF parameters. Change the value of these parameters from 8192 to 64000. These parameters can be changed by following these steps. Issue the CFGTCP command on the AS/400 command line and press enter. On the Configure TCP/IP screen, select option **3** (Change TCP/IP attributes). This brings you to the Change TCP/IP Attributes screen (see Figure 23). The parameters you change are TCP receive buffer size (TCPRCVBUF) and TCP send buffer size (TCPSNDBUF). Change them both to 64000.
Note: Setting these buffer sizes can cause congestion on the network in situations where you have hubs or gateways that cannot handle the same sizes of buffers. For these settings to be effective, you have to know your network and how to size these buffers to avoid congestion.	

Changing the maximum transmission unit is done by using the Change TCP/IP Interface (CHGTCPIFC) command, as shown in Figure 21.

```
                    Change TCP/IP Interface (CHGTCPIFC)

Type choices, press Enter.

Internet address . . . . . > '9.5.92.13'   Character value
Line description . . . . . .  TRNLINE      Name, *SAME, *VIRTUALIP...
Subnet mask  . . . . . .  .   '255.255.255.128'
Associated local interface . . .  '*NONE'
Type of service  . . . . . .  *NORMAL      *SAME, *MINDELAY...
Maximum transmission unit  . . .  *LIND     576-16388, *SAME, *LIND
Autostart  . . . . . .  .  .  *YES         *SAME, *YES, *NO
PVC logical channel identifier    *SAME     001-FFF, *SAME, *NONE
              + for more values
X.25 idle circuit timeout  . . .  *SAME     1-600, *SAME
X.25 maximum virtual circuits  .  *SAME     0-64, *SAME
X.25 DDN interface . . . . . .  .  *SAME    *SAME, *YES, *NO
TRLAN bit sequencing . . . . . .  *MSB      *SAME, *MSB, *LSB



                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 21. Change TCP/IP Interface

If the line used is a Token-Ring line, the maximum frame size can be modified using the Change Line Desc (Token-Ring) (CHGLINTRN) command, as shown in Figure 22.

```
                Change Line Desc (Token-Ring) (CHGLINTRN)

Type choices, press Enter.

Line description . . . . . > TRNLINE       Name
Resource name  . . . . . .   CMN05         Name, *SAME, *NWID, *NWSD
Online at IPL  . . . . . .   *YES          *SAME, *YES, *NO
Vary on wait . . . . . .     *NOWAIT       *NOWAIT, 15-180 seconds
Maximum controllers  . . . .  40           1-256, *SAME
Line speed . . . . . .       16M           *SAME, 4M, 16M, *NWI
Duplex . . . . . .           *HALF         Character value, *SAME...
Maximum frame size . . . . . > 16393       265-16393, *SAME, 265, 521...
Activate LAN manager . . . . .  *YES        *SAME, *YES, *NO
TRLAN manager logging level  . .  *OFF      *SAME, *OFF, *MIN, *MED, *MAX
TRLAN manager mode . . . . . .  *OBSERVING  *SAME, *OBSERVING...
Log configuration changes  . . .  *LOG      *SAME, *LOG, *NOLOG
Token-ring inform of beacon  . .  *YES      *SAME, *YES, *NO
Local adapter address  . . . . .  *ADPT     400000000000-7FFFFFFFFFFF...
Functional address . . . . . .  *SAME       *SAME, *NONE, C00000000001...
              + for more values
                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 22. Change Line Description (Token Ring)

Changing the TCP/IP buffer size is done by using the Change TCP/IP Attributes (CHGTCPA) command, as shown in Figure 23.

```
                Change TCP/IP Attributes (CHGTCPA)

Type choices, press Enter.

TCP keep alive . . . . . .   120          1-40320, *SAME, *DFT
TCP urgent pointer . . . . . *BSD         *SAME, *BSD, *RFC
TCP receive buffer size  . . . . 64000    512-8388608, *SAME, *DFT
TCP send buffer size . . . . . 64000      512-8388608, *SAME, *DFT
UDP checksum . . . . . .     *NO          *SAME, *YES, *NO
Path MTU discovery:
  Enablement . . . . . .     *YES         *SAME, *DFT, *NO, *YES
  Interval . . . . . .       10           5-40320, *ONCE
IP datagram forwarding . . . . . *NO      *SAME, *YES, *NO
IP source routing  . . . . . *YES         *SAME, *YES, *NO
IP reassembly time-out . . . . .  10      5-120, *SAME, *DFT
IP time to live  . . . . . .  64          1-255, *SAME, *DFT
ARP cache timeout  . . . . . .  15        1-1440, *SAME, *DFT
Log protocol errors  . . . . . .  *YES    *SAME, *YES, *NO


                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 23. Change TCP/IP Attributes

Some OS/400 performance considerations

For V4R4, enhancements for the performance of threaded applications have been made. If you do manual performance tuning of your system (the system value QPFRADJ is set to 0 or 1), you may have to tune your system again after V4R4 has been installed to avoid performance degradation when running Domino for AS/400. If the system value for QPFRADJ is set to 2 or 3, the system makes

the appropriate adjustments to keep Domino running smoothly on the new release of the AS/400 system.

If doing manual tuning of your system, you may need to increase the Activity Level (ACTLVL) of your system, which is the maximum number of threads that can run in your storage pools. You can change the ACTLVL using the Change Shared Storage Pool (CHRSHRPOOL), Work with Shared Pool (WRKSHRPOOL), or the Work with System Status (WRKSYSSTS) commands. The shipped default for Domino for AS/400 is to use the *BASE pool. The activity level associated with this pool may need to be increased. If you modified what pool the Domino for AS/400 is to use, you may need to increase the activity level of the pool you specified to insure good performance.

A partial list of the most common symptoms that might occur which would indicate that the system needs to be tuned again are:

- An excessive number of Domino clients time out or are dropped.
- The number of SERVER jobs in the Domino subsystem keeps growing.
- An excessive number of threads in a SERVER job have a status of MTXW and that status rarely seems to change for a thread. On the Work with System Status (WRKSYSSTS) screen, the Wait->Inel column is non zero. For more information on adjusting the performance on your system, see the article *Performance Tuning of AS/400 for Highly Threaded Applications*. You can find this article at:
  `http://www.as400.ibm.com/tstudio/tech_ref/perftune/Threads.htm`

For more information on other threaded applications that may require the system to be tuned, see the Informational APAR II11476, which is also referenced in the memo to users for V4R4.

Note

Authorized Program Analysis Report (APARs) are requests for a correction of a defect in a current release of an IBM-supplied program. An Informational APAR provides information not linked to a particular problem, such as considerations about a product or function, or specific technical information that was not previously documented in a manual.

The IBM AS/400 Authorized Problem Analysis Reports database can be accessed at `http://as400service.ibm.com`

On the left pane, click **Tech Info & Databases** and then **Software Problems - APARS**. A full text search is provided.

Other references are:

- *AS/400 Performance Capabilities Reference*, SC41-0607 (see the “Domino for AS/400” chapter)
- *Lotus Domino for AS/400: Performance, Tuning, and Capacity Planning*, SG24-5162

2.3.1.4 Conclusion

The Domino Web server provides advanced functionality for interactive applications that change dynamically and involve collaboration and workflow with the Web community. This is done through Domino databases. If you want to publish a static Web site, we recommend that you use the AS/400 HTTP server. Static home pages perform better than a Domino server because there is no data conversion required.

If you need to implement both types of usage, the Domino HTTP server is the solution. It can store static pages and objects stored in the AS/400 file system as well as documents and objects stored in Domino databases.

The AS/400 system is a very robust and reliable Domino server. By implementing the performance tuning recommendations in this section, you can fully use the performance advantage that the AS/400 system has to offer for Domino Web serving applications.

2.4 Choosing an AS/400 server

Which AS/400 server to choose depends on the environment that you are going to implement. What is the expected workload? Is the Domino going to run on a separate server, or is it going to run on an existing server already running some traditional AS/400 workload?

You can choose from the following options:

- Separate server:
  - AS/400e Dedicated Server for Domino (AS/400 DSD)
  - Traditional AS/400e server
- Existing server:
  - Traditional AS/400e server
  - Estimating the extra processing power you need

First, we offer a brief positioning of the two servers, AS/400e Dedicated Server for Domino and AS/400 Traditional Workload server, from which you should be able to decide whether the AS/400 DSD will be suitable for your implementation. The AS/400 DSD is very attractive when it comes to the price performance features.

Second, we introduce to how to estimate the needed processing power or how to size the AS/400 server that is required.

2.4.1 AS/400e Dedicated Server for Domino

The AS/400e Dedicated Server for Domino is the first server built especially for running Lotus Domino R5.0. It provides the reliability, manageability, and availability on which AS/400 has built its reputation. What's new is the specially low-cost-to-own, low-cost-to-buy package designed specifically for Domino applications.

The AS/400e Dedicated Server for Domino is best suited for:

- Existing Domino customers who want a more reliable, manageable server (see 1.3.1, “Reliability and availability” on page 13, and in 1.3.5, “Total cost of ownership” on page 18)
- Existing AS/400 customers who want to deploy a new server for pure Domino work, for example, Domino extranet, Domino Applications, Domino Messaging, Calendaring & Scheduling
- Existing Domino customers who want to consolidate several Domino PC servers into fewer footprints
- Any customer who wants a highly reliable, manageable Domino server, especially for a mixed Domino workload (for example, e-mail, plus Domino applications, plus Domino Web serving)
- Any customer who wants to deploy reliable, manageable Domino servers in one or multiple remote locations or departments

Good reasons to choose the AS/400e Dedicated Server for Domino are:

- **A first**: No other platform offers servers are designed just for Domino.
- **Upgradability**: From processor feature 2407 for mail and light applications, to processor feature 2408 for heavier mail and heavier applications plus more users, to processor feature 2409 for more robust Domino or for growth and capacity.
- **Domino application portfolio**: Hundreds of Domino applications are already ported to the AS/400 system. Search for them on the Web at: `http://www.as400.ibm.com/developer/domino`
- **Multiple workloads**: Run a mixture of Domino mail, Domino application serving, and Domino Web serving reliably on a single AS/400 server. This is compelling for small and medium-sized customers, where the alternative is to install (and manage) separate servers for each Domino function for acceptable availability.
- **Simplified management**: One server to manage easily, locally or remotely. A variety of management options and interfaces. Many tasks can be automated.
- **Up-time**: A special AS/400 “watchdog” job automatically stops and restarts a Domino server that begins to fail, without affecting any other work on the AS/400 system. If “Domino clustering” is configured, Domino switches users to another available Domino server.
- **Skills**: Existing AS/400 customers can leverage existing AS/400 skills. Domino may be backed up and managed with AS/400 commands and knowledge.
- **Speed of deployment**: The AS/400 system has a well-deserved reputation of being easy to get up and running quickly.

If you are thinking about Domino Mail, Domino Web Server, Domino Application Servers, Domino Backup Server all managed on *one* footprint, you should consider the AS/400e Dedicated Server for Domino.

2.4.2 Traditional AS/400e server

The traditional AS/400e server is the premier Business Application Server for small, medium, and large businesses. It's especially suited for running mixed

workloads (ERP, Domino, file and print serving, Java and Web serving) at the same time on the same server. Built for business, the AS/400e traditional workload server is known for reliably running mixed workloads while still maintaining its remarkable 99.9+% availability.

The AS/400e traditional workload server is best suited for:

- Any customer requiring the use of DECS (Domino Enterprise Connection Server), which is part of Domino R5.0, for real-time data movement to DB2
- Existing AS/400 customers who want to add Domino to an existing traditional AS/400e server
- Any customer who wants to run Domino applications and other non-Domino applications on the same server (one server instead of several servers to manage and maintain)
- Any customer who wants significant integration between Domino and relational data
- Existing Domino customers who want to consolidate a large user population from other Domino servers
- Any customer who, looking to the future, wants the broad scalability that the traditional AS/400e servers offer (this is often important for customers using Domino for Internet applications where volumes and growth are impossible to predict)

Good reasons to buy a traditional AS/400e server are:

- **Scalability**: Start with a few users and grow to thousands on a single AS/400e server.
- **Workload consolidation**: A large AS/400e can run up to 30 separate Domino server instances, plus AS/400 business applications and DB2 UDB for AS/400. No more server farms to manage. Fewer servers means lower costs.
- **Up-time**: The AS/400 system is the most reliable non-clustered server at 99.97%. Plus an AS/400 exclusive watchdog program makes sure a failed Domino partition is restarted automatically!
- **Seamless integration**: Domino for AS/400 provides integration with AS/400 Universal Database, Java, security, e-mail, backup/recovery, administration etc. No third-party tools required.
- **Simplified management**: One server to manage, locally or remotely. A variety of management options and interfaces. Many tasks can be automated.
- **Reduced cost of ownership**: Data integration and centralized management. Single hardware platform to manage both business and Domino applications.
- **Tightest integration with AS/400 Line of Business applications**: It makes sense to run Domino where the data is for easier, more efficient integration.
- **Mixed workload support**: The unique subsystem architecture of the AS/400 system logically isolates different applications from each other. You can safely and reliably run ERP, BI, e-commerce and multiple Domino partitions for e-mail, Web serving, etc. on a single AS/400 system.

If you are thinking about Domino Applications plus AS/400 applications, Web application serving, DB2 UDB for AS/400 server all managed on the same footprint, you should choose an AS/400e traditional workload server.

2.4.3 How far you can go with the AS/400e Dedicated Server for Domino

What was specified in the previous section for the two types of servers is what has been the message since the announcement of the AS/400 Dedicated Server for Domino (DSD). But there is a wider perspective for the AS/400 DSD. Potential workloads for the AS/400 DSD fall into three areas, which are explained in the following sections.

2.4.3.1 Appropriate

Our testing shows that pure Domino applications and most Domino applications that serve as a frontend to legacy data on another AS/400 system fall within the design guidelines for the Dedicated Server. You can expect that they will take full advantage of the price/performance characteristics of the Dedicated Server. These applications include:

- Domino e-mail
- Domino applications or agents using only Domino databases
- Domino.Doc
- Domino applications that use integration methods (for example, @DB functions, LS:DO, DECS) to access DB2 UDB for AS/400 databases *on another AS/400 server*. These integration methods take advantage of the AS/400 Distributed Relational Database Architecture (DRDA) support.

  Note: Some exceptional processing with DRDA might fall outside the recommended limits.

In addition, applications on the Integrated Netfinity Server, which share AS/400 resources (such as disk storage, printers, and tape devices) but use minimal AS/400 processing power, are also appropriate for the Dedicated Server. Both Windows NT-based file and print serving fall in this category.

2.4.3.2 Not appropriate

Applications that fall primarily outside the realm of Domino use excessive non-Domino capacity and generally do not perform well on the Dedicated Server. These applications include:

- Interactive 5250 applications (although sufficient interactive capacity is available to use a single 5250 session for systems administration functions).
- Standalone Java applications.
- Applications that access Domino databases without going through a Domino server (for example, using Domino APIs).
- Domino e-mail integration with other e-mail systems (for example Internet mail) through the AS/400 Anymail Framework. For e-mail integration on the Dedicated Server, you should use the built-in Domino R5.0 SMTP capabilities.

2.4.3.3 It depends

The design of the Dedicated Server includes a limited amount of non-Domino capacity, which is intended for complimentary work such as database integration or file and print serving. When your proposed workload includes this type of non-Domino work, the performance results you experience will depend on both the mix of work on your Dedicated Server and on your application design.

When you run an inappropriate workload (excessive non-Domino work) on a Dedicated Server, you can expect the following symptoms:

- With excessive interactive work (above the specified interactive CPW), overall performance of all work on the Dedicated Server degrades until the interactive work decreases.
- With excessive client-server or batch non-Domino work (above the specified processor CPW), overall performance of all work on the Dedicated Server may be impacted by increased processor utilization due to increased system overhead. However, new Domino work can be added in this environment and will replace the system overhead without increasing processor utilization. This assumes that you have configured your system correctly with Domino work running at a higher priority.

Several remedies are available for a workload that is not appropriate for a Dedicated Server configuration:

- Upgrade to a larger Dedicated Server with a higher processor CPW rating.
- Move the non-Domino work to a traditional AS/400e server.
- Increase the ratio of Domino work compared to non-Domino work. The Dedicated Server is designed to allow new Domino work to “reclaim” the CPU cycles being consumed by the system overhead that non-Domino batch work generates. By running more Domino work, you take better advantage of the CPU cycles available on the server.

Keep in mind that in general, the traditional AS/400e server is the best choice to provide real-time integration between Domino applications and DB2 UDB for AS/400 data on the same server.

The white paper “Evaluating Appropriate Workloads for the AS/400e”, explores the behavior and performance of light data integration on the same server to help with your planning and analysis. You can find the white paper at:
`http://www.as400.ibm.com/whpapr/dsd.htm`

However, our bottom line recommendation for data integration is either to use a Dedicated Server as a frontend to another AS/400 system or to run Domino applications on a suitably configured traditional AS/400e server that houses your DB2 UDB for AS/400 data.

It can also be expressed like this. The Dedicated Server treats processing as Domino work when the processing runs within the Domino R5.0 code. Processing that runs any function other than Domino R5.0 code should be considered as non-Domino work. Non-Domino work includes functions that a Domino thread or process calls, such as a LotusScript call to SQL. The processing time in the SQL code is non-Domino work. In other words, when an application goes “outside Domino”, it is performing non-Domino work.

The only definitive method for ensuring that your proposed application integration will fall within the desired performance guidelines for the Dedicated Server is to test. However, Figure 24 on page 54, provides general guidance to help you determine whether your implemented workload is considered to run within the Domino R5.0 code and, therefore, worth the effort to test it.

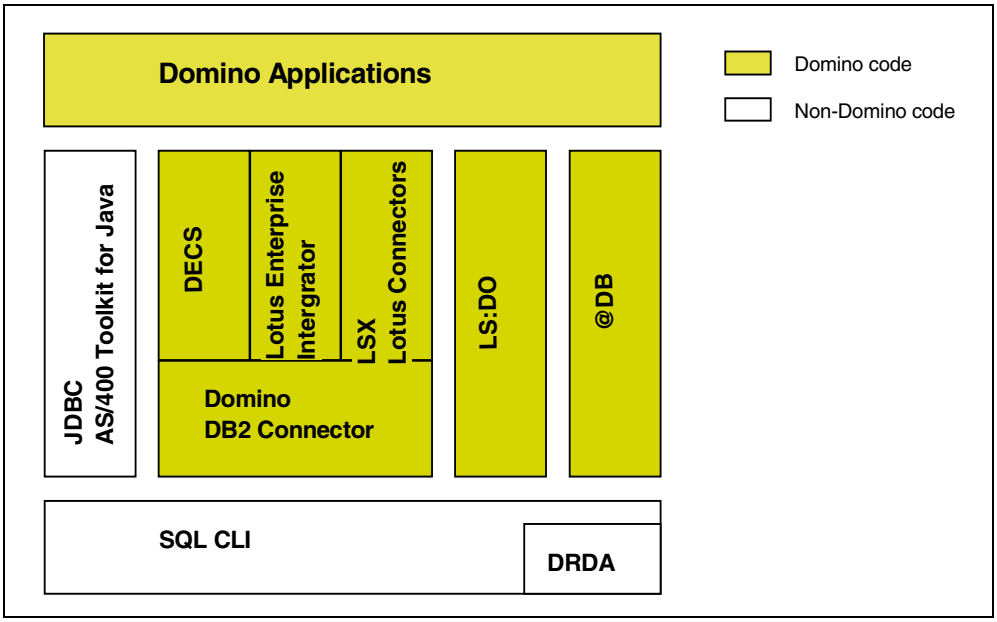


Figure 24. Distinguishing between Domino code and non-Domino code

2.4.4 Sizing considerations

Choosing the correct configuration for your Domino for AS/400 server is important for your success. We have a tool and two documents to help you size your system correctly:

- The IBM Workload Estimator for AS/400 is a Java-based tool for sizing an AS/400 server for a mixed workload, including Domino. This tool is available on the Web at: `http://as400service.ibm.com/estimator`

  The IBM Workload Estimator for AS/400 is an easy-to-use, browser-based workload estimating tool for use by customers, IBM sales and marketing personnel, and Business Partners. Accessing this tool via a Web browser provides access to the latest sizing information and system recommendations on the server being accessed. Both upgrades to existing AS/400 systems, and sizing a new system are supported by the Workload Estimator.

  Recommendations provided by the Workload Estimator are based on the definition of your Lotus Domino workload. The user interface provided by the Workload Estimator guides you through the process of sizing an AS/400 system as a Lotus Domino server. This process starts with the number of users you have and the type of work they will do. It also includes how much memory and disk space is needed to support these users. Finally, based on your input, the Workload Estimator selects the system for running your Lotus Domino workload.

- For sizing information geared toward basic uses of Domino, such as e-mail and out-of-the-box template applications, such as discussion databases, refer to the document *Sizing Domino for AS/400*, which is available at: `http://www.as400.ibm.com/domino/domsz2.htm`

  This is a large document with worksheets and information about every AS/400 processor. We suggest that you print it and keep it for your records when you have completed the worksheets.

- For sizing information about more advanced Domino applications, refer to the document *Domino for AS/400: Application Sizing Examples* at:
  `http://www.as400.ibm.com/domino/D4appsz.htm`

  It describes the performance characteristics of several sample applications. You can use the information in this document to help you choose the correct AS/400 processor for custom-written or purchased Domino applications.

Another way of sizing the AS/400 server required to run your Domino applications is to use the Capacity Planning Tool, BEST/1, on the AS/400 system. To use this tool, you need to know the workload size for the Domino application and enter it manually into the BEST/1 tool. Or, if your application is ready and can run a simulation test of the workload while running the performance monitoring on the AS/400 system, you can have the AS/400 system collect the necessary data for the workload and use the BEST/1 tool to create the workload base to be used when estimating the processor of the AS/400 system.

You can read more about BEST/1 in the redbook *Lotus Domino for AS/400: Performance, Tuning, and Capacity Planning*, SG24-5162.

Chapter 3. Managing directories and authentication

What is authentication? It is verifying that the users are who they say they are.

It consists in verifying such credentials as:

- Name
- Password
- SSL Certificate
- Cookie

These credentials are verified against an authority, such as a directory or SSL key ring.

Directories are the basis for user authentication and the cornerstone for security.

This chapter discusses what directories are available for Domino, how authentication is performed, and how the Domino directory infrastructure coexists with the AS/400 user profile management.

3.1 Domino Directory

The Domino Directory, in which previous releases referred to as the Public Address Book or Name and Address Book, is a database that Domino automatically creates on any Domino server.

The Domino Directory serves two purposes. It is a directory of information about users, servers, groups, and other objects that you may include in the directory yourself, for example printers. It is also a tool that administrators use to manage the Domino system.

For example, administrators create documents in the Domino Directory to connect servers for replication or mail routing, to register users and servers, to schedule server tasks, and so on. Typically, a Domino Directory is associated with a Notes domain. When you register users and servers in the domain, you create Person documents and Server documents in the Domino Directory. These documents contain detailed information about each user and server.

When you set up the first server in a Notes domain, Domino automatically creates the Domino Directory database and gives it the file name names.nsf. Additional users can be registered, and applications can be protected, as shown in Figure 25 on page 58.

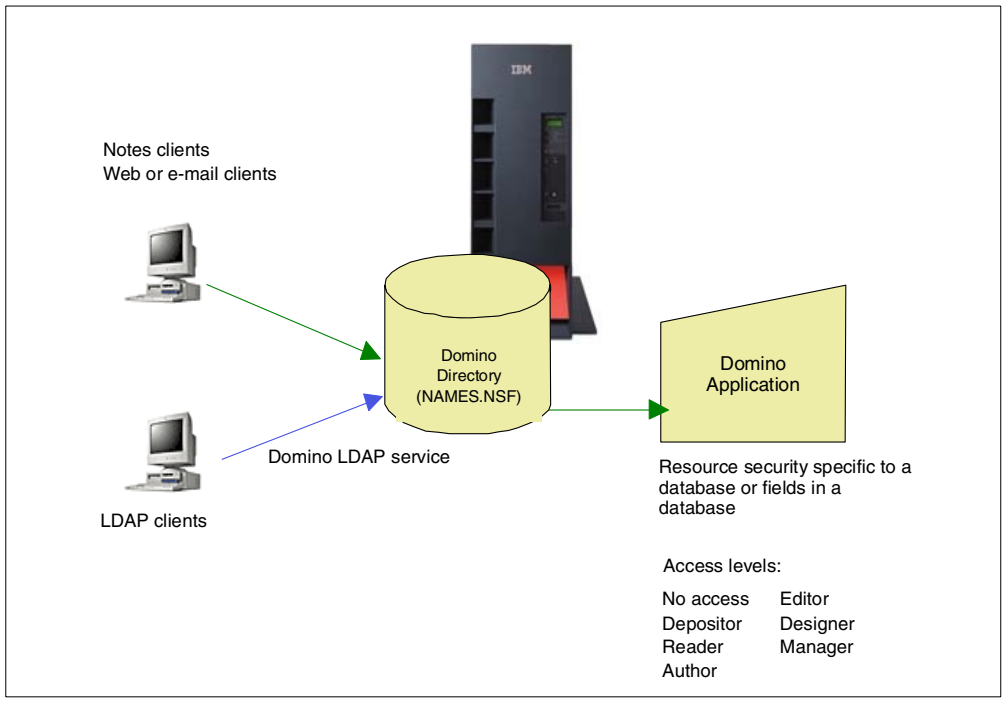


Figure 25. User registration: Single directory

When you add a new server to the domain, Domino automatically creates a replica of the Domino Directory on the new server. When you set up the first server in an organization, you create a Notes domain and a Domino Directory. Most organizations use one Notes domain and register all servers and users in one Domino Directory.

3.2 Domino Directory assistance

When you set up the first server in an organization, you create a Notes domain and a Domino Directory. However, there are reasons to use multiple domains and multiple Domino Directories (Figure 26).

Consider the example where, in a large organization, responsibility for system administration is distributed. In this case, creating separate domains and using separate Domino directories allows you to enforce security by setting up the ACL for each directory so that the OtherDomainServers group has limited access, for example, No Access or Reader Access.

Another reason to use multiple domains and directories is if your organization merges with another organization that uses Domino. In that case, you may decide to retain separate domains and separate directories, at least temporarily. To create an additional domain and Domino Directory, you perform a first server setup.

Some organizations create an additional Domino Directory that is not affiliated with a Notes domain. You may do this, for example, to manage non-Notes users, such as Web users. To create an additional Domino Directory, you create the directory using the PUBNAMES.NTF template. For example, if your company

sells e-mail addresses to other companies that use Domino, you may want to create a separate directory to hold the addresses.

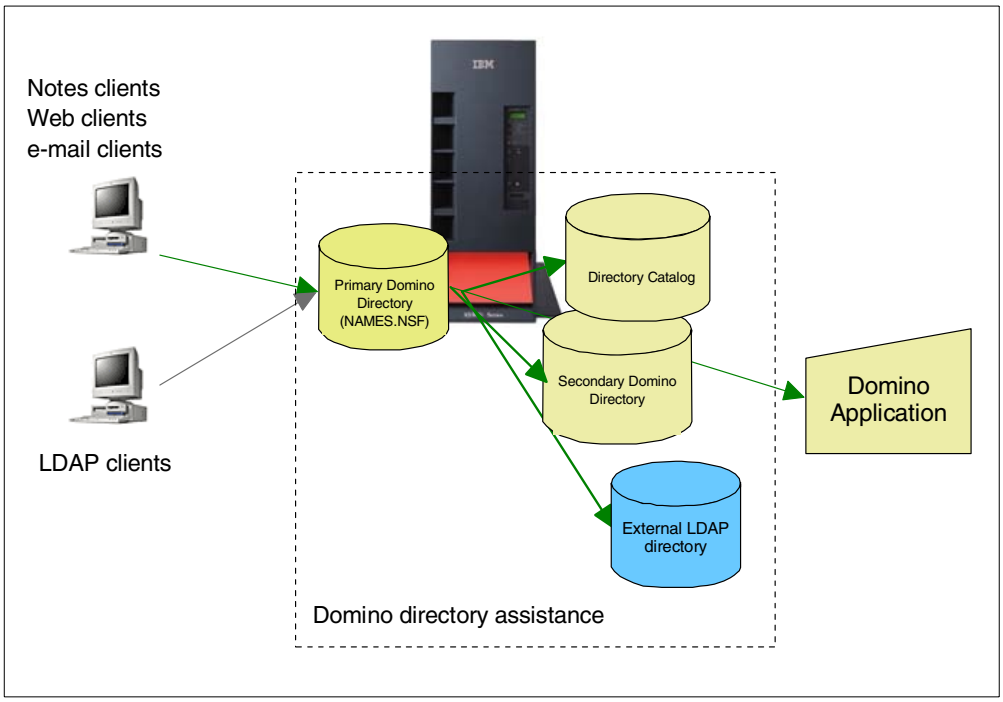


Figure 26. User registration: Multiple directories

In a system with multiple directories, the server's primary Domino Directory is the directory in which that server is registered. A server's primary Domino Directory uses the file name NAMES.NSF. From the standpoint of a given server, any Domino Directory within the organization that is not the server's primary Domino Directory is a secondary Domino Directory.

An organization may also use a Domino Directory along with a third-party LDAP directory. For example, an organization may have a Domino Directory on a Domino Web server, but use security credentials in a third-party LDAP directory to authenticate Web users that connect to the Domino server. If you run the Domino LDAP service, you can also refer LDAP clients that use the service to a third-party LDAP directory.

It's important to plan a strategy for managing multiple directories. You can set up a directory catalog and directory assistance to make it easy to extend name lookups and client authentication beyond the primary Domino Directory.

3.2.1 Directory catalog

A directory catalog consolidates the most frequently requested directory information from one or more Domino directories into a single, lightweight, quick-access database. To configure a directory catalog, you create a database from the template DIRCAT5.NTF. There are two types of directory catalogs: mobile directory catalogs and server directory catalogs.

Notes users store local replicas of a mobile directory catalog so they can quickly address mail to anyone in your organization, even when they're disconnected from the network. Type-ahead addressing automatically searches the mobile

directory catalog and reduces network traffic since name searching doesn't have to occur on the server.

Server directory catalogs expedite name searches in secondary Domino directories that occur on the server.

3.2.2 Directory assistance

Directory assistance is a feature that enables users and servers to locate information in a directory that is not a server's primary Domino Directory. You can set up directory assistance for secondary Domino directories and for LDAP directories, for example, for third-party LDAP directories.

Directory assistance extends Notes name lookups, LDAP searches, and Web client authentication to secondary Domino directories and to LDAP directories. To configure directory assistance, you use a Directory Assistance database created from the template DA50.NTF. In this database, you create a document for each directory you want to search. In the document, you specify, among other things, the directory location and naming rules that reflect the names in the directory.

To set up directory assistance in a Notes domain, complete these procedures:

1. Set up a Directory Assistance database.
2. Set up directory assistance for each secondary Domino Directory that you want to include in directory assistance.
3. Set up directory assistance for each LDAP directory that you want to include in directory assistance.

3.3 Authentication process

After you set up name and password access and create Person documents for Internet and intranet users in the Domino directory or additional directories, Domino authenticates users when:

- They attempt to do something for which access is restricted.
- Session-based authentication is enabled.
- Anonymous access is not allowed on the server.

For example, when a user tries to open a database that has an ACL with No Access as the default, Domino challenges the user for a valid user name and password. Authentication succeeds only if the user provides a name and password that matches the name and password stored in the user's Person document and if the database ACL gives access to that user. Anonymous users are not authenticated.

In addition to HTTP basic authentication, session-based authentication has been available since Domino Release 5.0.

3.3.1 HTTP basic authentication

HTTP basic authentication (Figure 27) employs user IDs and passwords. You have to configure the server to identify which parts of the document tree are protected. These zones of protection are known as *realms*. Each realm is associated with a set of user IDs and passwords that are allowed access. Realms can contain any kind of server object, such as CGI programs and HTML pages.

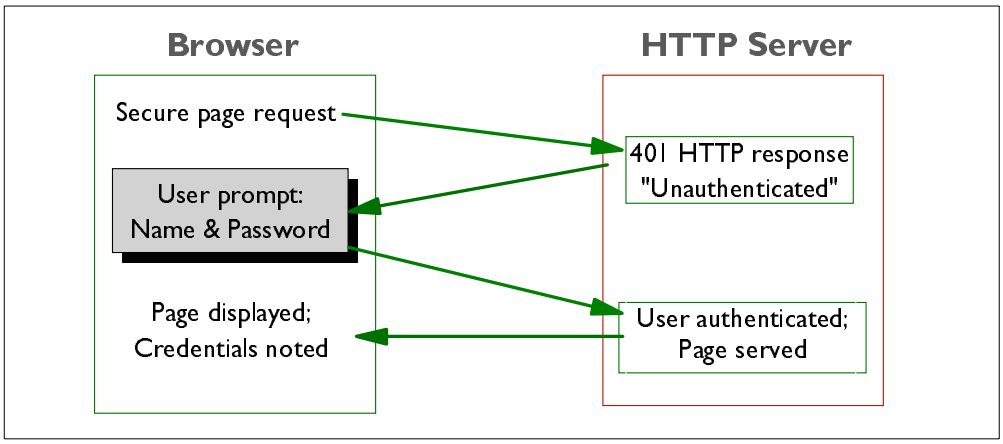


Figure 27. How basic HTTP authentication works

Basic authentication relies on a challenge mechanism to prompt users to authenticate themselves. When a client requests a URL, the server checks to see if the URL requires user authentication. If it does, the server rejects the request, with a status code of 401. Then, a dialog page appears on the user's display asking for a user ID and password. When the user provides them, the browser resends the original request, but with the addition of the following MIME element within the HTTP header:

```
Authorization: Basic <userID and password block>
```

The user ID and password block are constructed by creating a string of the form userID:password and then encoding it using the base64 algorithm.

You may wonder, given the above description, why you are not repeatedly prompted for a password every time you access a new restricted page. The reason is that the browser caches the user ID, password, server name, and realm name in memory. Then, if it receives another 401 status code for the same server/realm combination, it can re-issue the request using the appropriate user ID and password. In fact, most browsers go one stage further than this and send a user ID and password for any URL that is likely to need it.

Netscape Communicator and Microsoft Internet Explorer send the information with any URL that is in the same logical directory. For example, imagine that the user requests `http://host/secret/foo.html` and is prompted for a password with the realm name PRIVATE. Netscape Communicator stores the details in memory. Later, if the user clicks on a link to URL `http://host/secret/bar.html`, the browser sends the user ID and password for realm PRIVATE in the request, without waiting for a 401 challenge from the server.

The objective of these tricks is to reduce network traffic and improve responsiveness, by eliminating a number of invalid requests and 401 status responses. They also, unfortunately, have the side effect of re-transmitting the user ID and password when it may not, in fact, be necessary.

Web realm

Using a Domino Web realm, you can specify a string of text that is displayed when users try to access a certain drive, directory, or file on a Web server. When the browser prompts the user for a name and password, the browser's

authentication dialog displays the realm text string. The browser uses the realm to determine which credentials (user name and password) to send with the URL for subsequent requests. It caches credentials for different realms to avoid prompting the user again for the same credentials.

The realm string also applies to requests mapped to paths that have the specified path as their root, provided that the child paths of the root do not already have a specified realm. For example, the realm string specified for /LOTUS/DOMINO/DATA would also apply to a request mapped to /LOTUS/DOMINO/DATA/FINANCE if the latter did not have a specific realm specification.

If there is no realm specification for a given path, Domino uses the path from the request as a realm string.

3.3.2 Session-based authentication

A session is the time during which a Web client is actively logged onto a server. Session-based name-and-password security includes additional functionality that is not available with basic name-and-password security. You use the Server document in the Domino Directory to specify settings that enable and control session authentication.

For session-based authentication, “Log in” invokes the following actions:

- Occurs at authentication
- Creates unique session ID on that server
- Creates a browser cookie with a session ID

At “Log out”, the following actions occur:

- The session ID is invalidated on the server.
- The cookie is destroyed.

An example is shown in Figure 28.

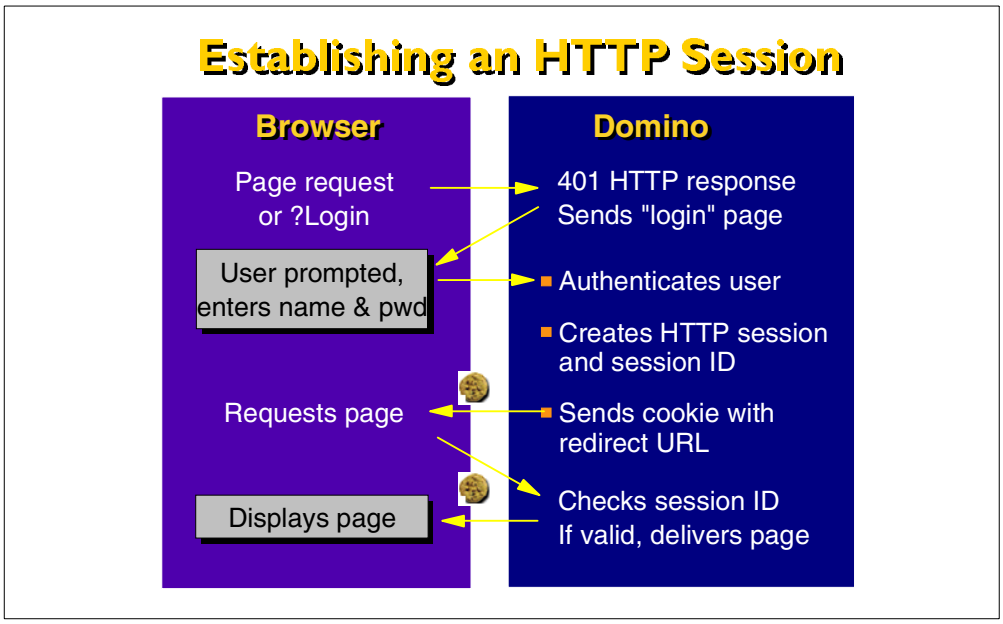


Figure 28. Session-based authentication: Establishing an HTTP session

To use session-based authentication, users must have a browser that supports the use of cookies. Domino can use cookies to track user sessions. Cookies are commonly used to track persistent data of a server or a specific Web application.

Note: If you turn cookies off in the browser, session-based authentication doesn't work.

The cookie that is created in the browser is deliberately set with no Expiration date/time (so that the cookie is destroyed when the browser shuts down) and no domain (so the cookie is only valid for the server that issued it).

The cookie name is DOMAUTHSESSID. It contains the encrypted user ID, plus encrypted random information. This prevents *spoofing*.

Session-based name-and-password authentication allows Web clients to access a server using a name and password. Session-based name-and-password authentication includes additional functionality that is not available with basic name-and-password authentication. With session-based name-and-password authentication, you can customize the log-in form, specify a default log-out time period, and specify the maximum number of user sessions. You specify the customized log-in form in the Domino Web Server Configuration database.

There are several advantages using HTTP sessions, for example:

- Security
  - Credentials are only sent “in the clear” once (unless SSL is used, which means credentials are never sent “in the clear”)
  - A user can “log out”.
  - A session can be timed out.

- Ease of use
  - The user is only prompted once for credentials.
  - A realm definition is not needed.
  - Active user sessions can be seen at the server console, using the command:

    ```
    tell http show users
    ```

  - Login forms can be customized.

Note

If your servers are set up for round-robin DNS, do not use session-based name-and-password authentication. The servers cannot store the session information in memory when using round-robin DNS.

In addition, if a server is rebooted or crashes, session information is lost. The user must re-enter the name and password.

3.3.3 Considerations about basic and session-based authentications

There are two obvious loopholes in HTTP basic authentication and session-based authentication:

- The user ID and password are included in the packet header, which means that they can be captured by anyone with a network sniffer or trace tool at any place in the session path.
- The user ID and password are cached in the browser so, if you leave the machine unattended, anyone can use your ID to access restricted information.

The second loophole is no different from any other situation where a machine is left unattended. The solution is one of user education: Always lock the screen when you leave your desk. Note that the caching is in memory, so the user information is lost once the Web browser has been shut down.

The first loophole is more significant. The user ID and password are not encrypted when they are placed in the packet header, but instead are encoded with base64. Base64 is an algorithm that forms part of the Multipurpose Internet Mail Extensions (MIME) protocol. It is a mechanism that turns any bit stream into printable ASCII characters (it is described in RFC1521). In fact, the objective of base64 is not for masking data at all, but to provide a method to send binary data through a mail gateway that can only handle character data.

The result of this is that by capturing the Authorization: Basic header from an HTTP request, an attacker can easily extract the user ID and password.

How serious is this exposure? Within a corporate network, it may not be a big problem. In fact, base64 offers protection of user IDs and passwords that is superior to many older protocols that send them as clear text. On the Internet, it is a different story. Here you have to assume that someone, somewhere, is tracing everything you send. Clearly HTTP basic authentication should not be used as the sole method of protection for any critical resource.

How can you solve the problem of sending confidential information like user names, passwords, and credit details over the network securely? The solution is to ensure that the basic, or session-based, authentication, and subsequent communications are using an encrypted connection in which it can operate. Secure Sockets Layer (SSL) is a good example of a protocol that encapsulates HTTP data in this way. Domino SSL implementation is described in Chapter 4, “Domino SSL implementation” on page 71.

3.3.4 Anonymous Internet and intranet access

When you set up anonymous access, Internet and intranet clients can access servers without identifying themselves. Domino does not record these clients' database activity, for example, in the log file and in the User Activity dialog box.

With anonymous access, you never know who is accessing databases on the server. Therefore, you cannot use the client's identity (that is, the client's name and password) to control access to databases and design elements. Use anonymous access when you do not need to know who is accessing the database or when you do not need to control access based on client identity.

You can use anonymous access with TCP/IP or SSL on any server that runs NNTP, LDAP, HTTP, SMTP, or IIOP. For each Internet protocol enabled on the server, you can specify the method of security. For example, you can enable SSL for HTTP connections, but require name-and-password authentication for NNTP connections that use TCP/IP.

3.4 Coexistence with AS/400 security

Domino security and the underlying AS/400 security work together to achieve two objectives:

- The portability of Domino applications:

  Access control for Domino for AS/400 must work just like it does on other Domino server platforms.

- The security and integrity of other applications that might share the same AS/400:

  Domino for AS/400 must follow the security rules that apply to every other AS/400 application.

3.4.1 AS/400 security

The OS/400 operating system includes integrated security. OS/400 security has four major components:

- **A system-wide security policy**: You can set AS/400 system values to establish security policies that affect all AS/400 users. For example, you can specify what the system does when an interactive AS/400 session is inactive for a period of time. You can specify composition rules for AS/400 passwords, such as requiring that passwords contain at least one digit.

- **User profiles**: Every AS/400 user has a profile that defines the user's security and environment characteristics.

  For example, the special authorities value in a user profile specifies whether the user can perform system functions such as controlling jobs or setting up

new users. The output queue value in a user profile specifies the default location for the user's print requests.

Not all Domino for AS/400 users need an AS/400 user profile.

- **Resource security**: For every object on an AS/400 system, you can specify which users have authority to access the object. Conceptually, AS/400 resource security is similar to access control lists in Lotus Notes.
- **Security auditing**: The AS/400 system has the ability to audit security-relevant events that occur on the system and to audit access to specific objects. Security auditing is defined with system values. Audit entries are captured in a security audit journal and journal receivers.

For more information about AS/400 security, see *Tips and Tools for Securing Your AS/400*, SC41-5300.

3.4.2 What happens when the Domino software is installed

AS/400 objects, such as programs (*PGM) and service programs (*SRVPGM), are put in the AS/400 QNOTES library. Files that contain symbolic links to the programs, service programs, and other objects in the QNOTES library are created in the /QIBM/PRODDATA/LOTUS/NOTES directory in the AS/400 integrated file system. These symbolic links provide access to the objects in the QNOTES library from the integrated file system.

The AS/400 integrated file system provides a directory structure similar to PC operating systems (DOS, Windows) or UNIX operating systems.

The /QIBM/USERDATA/LOTUS/NOTES directory is created in the integrated file system. Users who create programs to be accessed by Domino must add symbolic links to their programs in this directory.

The QNOTES user profile is created for use by Domino and Notes application programs that run on the AS/400 system. This user profile is intended for system functions, and for integration with the underlying AS/400 security mechanisms. Therefore, it does not have a password. Individual users cannot use the QNOTES user profile to sign on the AS/400 system.

3.4.3 Domino for AS/400 users

A Domino server on an AS/400 system can have three types of users:

- **Domino-only users**: These users connect to the Domino server from a Notes client or a Web browser. They do not use any AS/400 functions except the Domino for AS/400 server. These users do not need an AS/400 user profile.
- **Domino and AS/400 users**: These users connect to the server from a Notes client or a Web browser. They also access your AS/400 system in others ways, such as with a 5250 workstation emulation or Client Access. These users need both an AS/400 user profile and a Domino registration.
- **Domino users who need DB2 UDB for AS/400 access or access to AS/400 applications**: These users might never appear to run an AS/400 application. However, they may use a Domino application that accesses DB2 UDB for AS/400 data. Whether they need an AS/400 user profile depends on how you design your Domino application.

3.4.4 Connecting to the AS/400 system and Domino for AS/400

To access any AS/400 client/server or 5250 application, the user needs an AS/400 user profile and password. When accessing a Domino application, the user either access the application as an “anonymous user”, or they must use a user identification and password.

The AS/400 user profile and the Domino user identification are managed in two different places, as shown in Figure 29:

- The AS/400 user profiles and password are managed in the OS/400 environment, either from AS/400 Operations Navigator or from a 5250 display.
- The Domino user and password are maintained in the Domino Directory or in any additional directory defined in the Domino server environment (as explained in 3.2, “Domino directory assistance” on page 60).

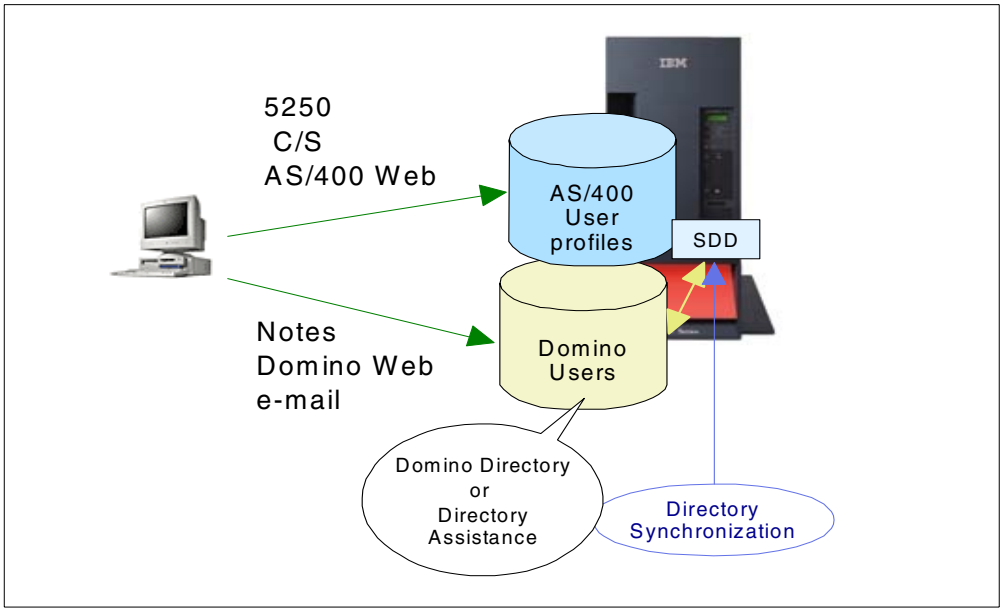


Figure 29. Coexistence with AS/400 security

Domino for AS/400 Single Logon, provided with Domino for AS/400, is a security tool that performs password synchronization for users of Microsoft Windows, Lotus Notes, and OS/400. This capability allows users to log on once and not have to separately log on to the Notes client or to AS/400 Client Access.

This tool can be helpful for any user that uses both a Lotus Notes client and AS/400 applications. The user can take advantage of changing their password in one place only.

More information about this tool can be found in the “Domino for AS/400 Single Logon” chapter in *Lotus Domino for AS/400 R5: Implementation*, SG24-5592.

3.4.4.1 System distribution directory and directory synchronization

The AS/400 system provides a directory that contains information about people (such as name, department, phone numbers, address, company, and electronic mail address) about users of the system itself and about users on other systems or servers. The information about users on other systems or servers makes it

easy to send them mail, using other e-mail functions available on the AS/400 system (OfficeVision/400, Send Distribution (SNDDST) command, SMTP/POP3, third-party tool). The AS/400 system keeps this user information in the system distribution directory (SDD).

Domino for AS/400 directory synchronization, which is included in Domino for AS/400, gives the ability to automatically synchronize the user information in a Domino Directory with the user information in the AS/400 system distribution directory. One advantage of directory synchronization is to avoid manually updating the user information in two places. You can think of directory synchronization as a form of replication that updates a Domino Directory from the AS/400 directory, the AS/400 directory from a Domino Directory, or both, based on your specifications. Directory synchronization is also similar to shadowing an AS/400 system distribution directory.

Understand that directory synchronization only updates the directory information. It does not automatically authorize users to either the Domino server or the AS/400 system. Also, it does not synchronize any password.

Domino for AS/400 directory synchronization can be very useful in any environment or e-business application, where e-mail will be largely exchanged between Domino e-mail users and any local or remote AS/400 user using one of the other e-mail functions available on the AS/400 system (OfficeVision/400, Send Distribution (SNDDST) command, SMTP/POP3, third-party tool).

3.4.5 Accessing DB2 UDB for AS/400 from Domino applications

A great feature of Domino for AS/400 is the integration between Domino and DB2 UDB for AS/400 databases. Several APIs and tools are described in 2.2, “Integration with AS/400 applications” on page 31. See Figure 30.

All methods for accessing DB2 UDB for AS/400, or calling an AS/400 stored procedure, from Domino establish a connection from Domino to the AS/400 system. The connection specifies both the user profile, whose authority the system uses to access DB2 UDB for AS/400 database files, and a password for that user profile.

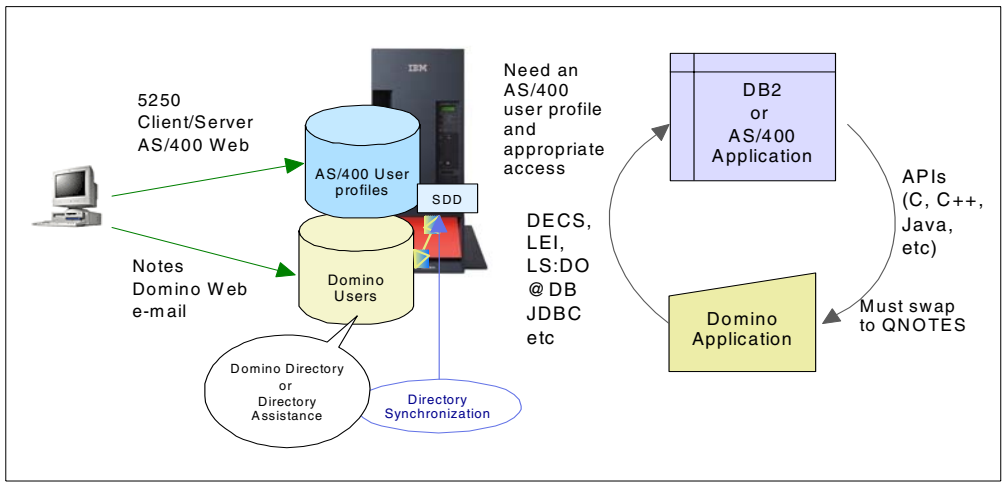


Figure 30. Security for Domino for AS/400 enterprise integration

A discussion about this topic is available in *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345, in the “Options for Integration with Enterprise Applications” chapter (in section “Security Aspects”).

In many places, the user profile and password do not need to be known by the Domino user. They can be hardcoded in the Domino application code (not recommended, even if the samples provided in this redbook are written that way), or maintained in a Domino database and accessed by the Domino application.

A drawback is that the access to the DB2 database is always performed using the same user profile. Therefore, there is no trace of who modifies data in DB2 UDB for AS/400 tables, unless additional code to implement a logging of database accesses is written.

3.4.6 Accessing Domino from AS/400 applications

When an AS/400 application must access Domino data or resources, the AS/400 application cannot work directly with Domino resources. AS/400 applications can retrieve (and potentially update) information in Domino databases. The following basic steps show you how to gain access to a Domino database from an AS/400 program:

1. To access a Domino database, an AS/400 program uses Domino application programming interfaces (APIs). These Domino APIs are available for C, C++, and Java programming languages. An RPG API is also available.

   More information about Domino for AS/400 APIs and APIs toolkits can be found in the “C, C++, RPG Application Programming Interfaces” chapter of *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345, and on the Web at: `http://www.as400.ibm.com/domino`

2. Domino processes the database request as a server program (not a client/server program). Therefore, the Domino user for the database request is the user who is associated with the server.

3. A Domino server can have more than one NOTES.INI file. The NOTES.INI file specifies the user for server jobs for that INI file.

4. Your AS/400 program can use an API to point to the INI file that is appropriate for the Domino databases that the program wants to access. If your AS/400 program does not explicitly specify an INI file, Domino uses the default user for the server.

5. The Domino user must have the necessary authority to the Domino database.

6. To gain access to the Domino server and the NOTES.INI file, the AS/400 program that contains the APIs must run under the QNOTES user profile. This program cannot use adopted authority because adopted authority does not work with the integrated file system, where the Domino databases reside.

   A user profile switching mechanism must be coded to switch from the current AS/400 user profile to the AS/400 user profile QNOTES, which is allowed to work with Domino resources. This switching mechanism must be implemented in each AS/400 program that accesses Domino resources.

   For your Domino server to function properly, the QNOTES user profile must have *ALL authority to all of the Domino objects. Therefore, you should avoid giving users authority to the QNOTES user profile, either directly or indirectly. Instead, use specific programs for specific functions and use profile swapping.

An example is provided in *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345, in the “Java and Domino” chapter.

Chapter 4. Domino SSL implementation

Implementing e-business often means that you will, at some level, exchange sensitive data between the clients and the servers. This is when security becomes an issue. How well funded Domino or Notes is when it comes to security will be covered in this section, where we reveal:

- The strong security infrastructure that has always been part of Notes and Domino
- How Notes and Domino R5.0 support today's open Internet security standard (X.509)

These elements are the basics in the Domino security infrastructure and cover the client security, the server security, and the transmission security when sending data between the clients and the servers, as shown in Figure 31.

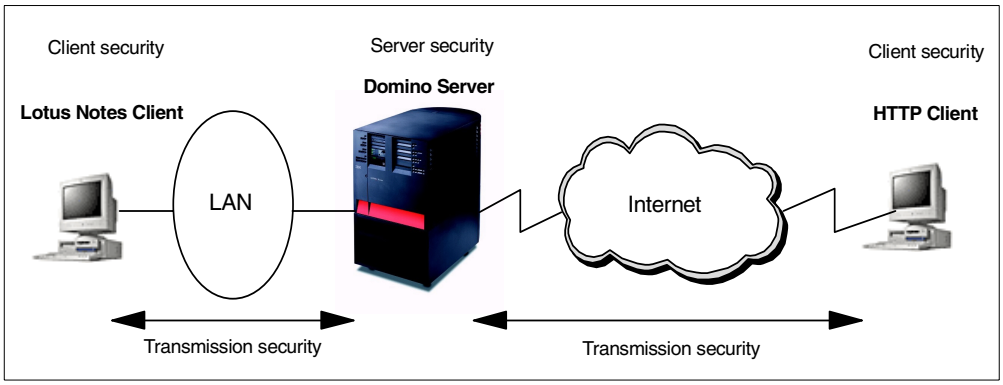


Figure 31. Domino security infrastructure

There are two types of clients that we cover: the traditional Lotus Notes client and a Web browser client. They are two totally different types of clients, but are authenticated by the same security system that has been implemented in Notes and Domino. The exception is the validation and the use of certificates when securing the transmission.

Lotus Notes clients and Domino has their own way of dealing with certificates compared to the way a SSL connection uses certificates when communicating between a Web server and a Web browser.

It is important that you understand how the Domino security works so you can implement the right level of security. We go through the basics of Domino security, followed by the basics of using secure connections between clients and servers.

When talking about security in Notes and Domino, you can look at it as nine layers. The upper layer is simply getting access to the network where the Domino server is placed. The lowest layer involves security at the field level within a Domino document. The nine layers, from top to bottom, are:

- Network
- Domino server
- User authentication
- Database

- Views/forms
- Document
- Section editor
- Hidden paragraphs
- Edit fields

We focus on the upper three layers, that is, security in Notes and Domino from an infrastructure perspective. However, to give you a complete picture, we briefly describe the other levels of access control. If you are interested in more information about security in Notes and Domino on the application level, refer to the “Securing Your Application” chapter in *Lotus Domino R5.0: A Developer’s Handboo*k, SG24-5331.

A section at the end of the chapter covers how Domino security coexists with the AS/400 security. This topic becomes very important as soon as you need to integrate Domino applications with AS/400 applications.

Chapter 9, “Sample project Domino server setup” on page 215, through Chapter 11, “Sample project SSL configuration” on page 265, take you step-by-step through how the security setup is done for this redbook’s sample project infrastructure.

4.1 Domino server access control

Server documents in the Domino Directory are used to control access to a Domino server. The implementation of security does not come for free.

Applying server access restrictions activates an additional security code that uses server processing cycles and can increase the time taken for a user to gain access to the server. However, access restrictions are an effective place to start with security controls. Using the restriction fields in the server documents, you can quickly restrict several types of access to the server. You update the server access control parameters through the Domino Administrator. The server document can be selected from the Communicator portion of the screen when you select the configuration tab in Domino Administrator as shown in Figure 32.

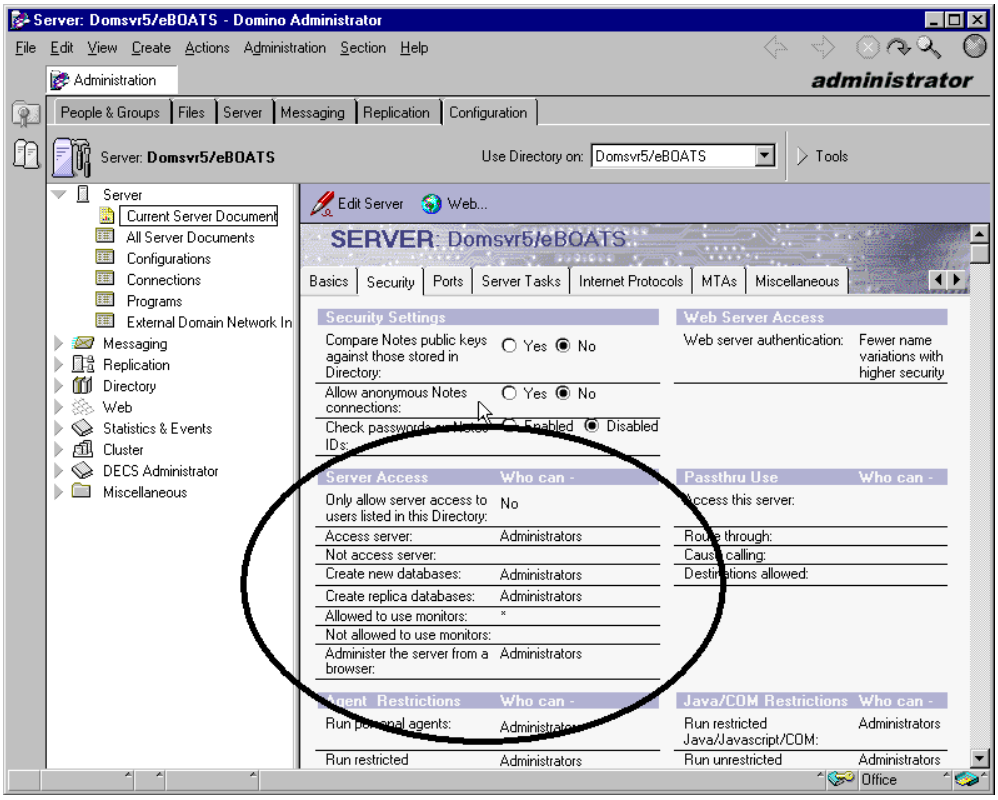


Figure 32. The Domino server document

Refer to the Domino Administrator documentation for an explanation of all the various security settings you can specify in the server document.

4.2 Domino data access security

After users or other servers gain access to a server, they want some level of access to the data held by that server. The database layer of the Domino security model and the layers below that deal with data access, each layer providing more granular control than its predecessor. In fact, the access controls mirror the way that Domino stores and presents data:

- **Database access**: Domino databases lie at the heart of the system. The records in a database are actually documents that have usually been entered by a user or administrator. Database access control facilities, therefore, provide the broadest control over who can do what to data on a Domino server.
- **Form access**: Documents are not free-form text, but are in fact filled-in forms. When designing forms, you can use form access controls to specify who has access to the contents of a database in more detail than you can by using the database access controls.
- **Document access**: Once a form is filled in to create a document, the owner of the document can further restrict access to it. The degree to which this is allowed depends on controls within the form from which the document is created.

- **Section access**: Many documents contain data of varying sensitivity. In practice, this means that you want to prevent certain users from updating or possibly even from reading parts of the document, but you want other users to have full access. One way to achieve this is to divide the form into sections and apply section access controls to it.
- **Field access**: This is the most granular form of data access control. It allows you to control access to individual fields on a form or document. In addition to specifying user access, field access controls can limit the treatment that data receives when it is transmitted or stored.

For more details on the data access security, see Appendix F, “Domino data access control” on page 391.

4.3 Notes client authentication and validation

The base on which all Notes security is built is user authentication. Authentication is important because it allows you to differentiate one user from another. Without it, you could not identify whether users are who they claim to be. It is, therefore, the key to providing restricted access to Notes resources.

The Notes authentication procedure depends on a certificate, an electronic stamp that indicates a trust relationship between the user and the server. The certificate is stored in a Notes ID file. Lotus Notes certification and authentication is a fairly complex process. Notes uses a number of cryptographic techniques, such as public key and symmetric key encryption, digital signatures, and public key certificates.

Confidentiality and integrity, as they apply to database replication and mail items in transit through the network, ensure that what arrives is identical to what was sent, and only the intended recipients can access the contents. All these features taken together constitute the security infrastructure of Notes.

4.3.1 Lotus Notes certification hierarchies

Lotus Notes authentication is based on Notes certificates, which are stored in Notes IDs. When a Lotus Notes user attempts to connect to a Lotus Domino server, the client and server present their certificates to each other. By examining certificates, the client identifies and authenticates the server, and the server identifies and authenticates the user.

4.3.2 Notes certificates

Casually speaking, a certificate is an electronic “stamp” that indicates a trust relationship among the entities in the Notes world. In actuality, a certificate is a digitally signed message added by a certifier to a Notes ID file.

The certificate contains:

- The certificate owner’s name and details
- The certificate owner’s public key
- The certifier’s name and details
- The certifier’s public key
- The certificate expiration date

The certificate is then certified, that is, digitally signed by the certifier using the certifier's private key, to prove its authenticity.

The certificate is then stored in a Notes ID file and the Domino Directory. A Notes ID file is a database that stores Lotus Notes certificates and private/public key pairs. The certificates registered to the Domino Directory are referred to by all users and servers belonging to the Domino domain, when they attempt to encrypt or digitally sign mail messages or document data. Note that the certificate itself does not contain any private information. Therefore, it is open to the public and can be distributed anywhere.

Figure 33 shows the flow for certifying a notes user.

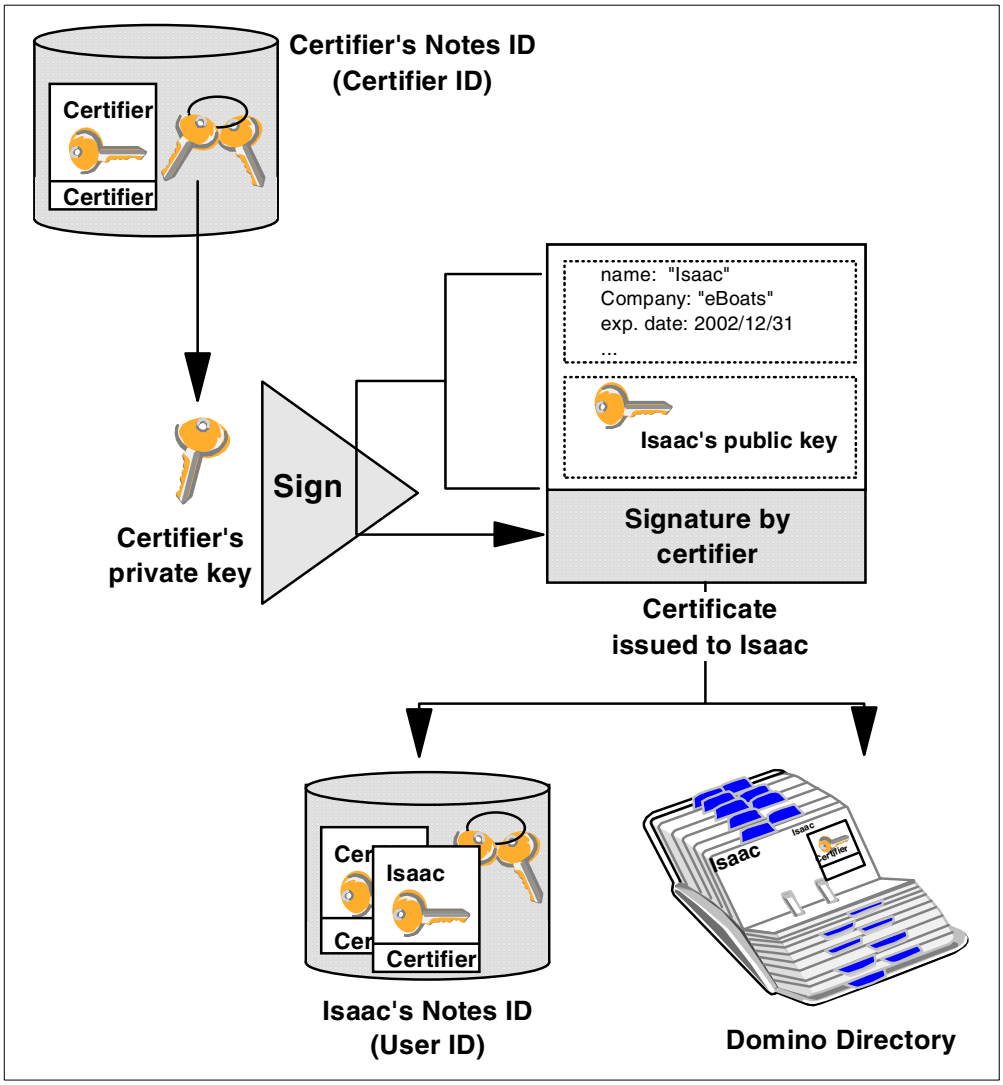


Figure 33. Flow for certifying in Domino

4.3.3 Using X.509 certificates

The Domino Server and Notes client in R5.0 add support for X.509 v3 certificates as well. The Notes client has the ability to request a certificate from any certificate authority, including a Domino R5.0 certificate authority, and store the X.509 v3 certificate in the Notes ID file.

4.3.4 Certification hierarchies

Lotus Notes used to provide two types of certification: flat and hierarchical. Organizations using flat names may use several certifier IDs. Each user ID and server ID can include separate certificates generated by each flat certifier ID. Organizations using hierarchical certification have one organization certifier and optionally up to four layers of organizational unit certifiers below.

4.3.5 Flat certificates

Lotus Notes/Domino R5.0 cannot create new flat IDs. However, flat ID maintenance is still supported by Notes/Domino R5.0 for compatibility with earlier versions. If your organization wants to use a flat ID with Lotus Notes/Domino R5.0, you must retain at least a prior version of Lotus Notes client to create a new flat ID.

New installations are encouraged to start with hierarchical names, and existing flat installations are encouraged to convert to hierarchical, because of the increased security and flexibility of access control, ID file generation and certification, and maintenance.

4.3.6 Hierarchical certificates

In hierarchical certification, an organization may be layered with the organization certifier at the top and up to four layers of organizational unit certifiers below. When users or servers are registered with a certifier, they receive a certificate signed by that certifier and inherit the certification hierarchy of the layers above.

For example, consider the certification hierarchy shown in Figure 34. This shows an organization named eBoats, subdivided into two organizational units, East and West. When Jennifer shows up on her first day of work, the administrator of East/eBoats defines her as a new user. One of the results of this process is a new, randomly-generated, RSA private/public key pair. The administrator then creates a certificate for Jennifer, by signing her new public key using the East/eBoats certifier private key. As a result, Jennifer's user ID inherits the certification hierarchy of the East/eBoats certifier.

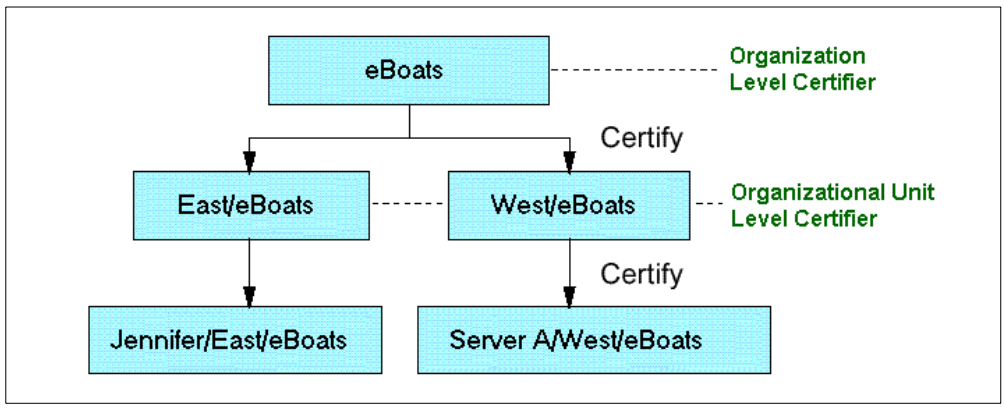


Figure 34. Hierarchical certificates

Users and servers in the organization have fully distinguished names based on their certifiers. Each layer in the certification hierarchy inherits the fully distinguished name of the certifier used to create it and is, in turn, an ancestor to

the layers below it. In this example, the organization level certifier eBoats has the fully distinguished name “O=eBoats,” an organizational unit level certifier East has “OU=East/O=eBoats,” and Jennifer has “CN=Jennifer/OU=East/O=eBoats.”

Users and servers may authenticate with each other if they have at least one common ancestral certificate. In our example, this means that all users in the organization can authenticate with each other because they have the eBoats certifier in common. Entities that don’t share at least one common ancestor can still authenticate by going through a cross-certification process.

4.3.7 Notes IDs and Domino Directory

Now let us see how this certification hierarchy is reflected in the Notes ID files and Domino Directory. When an administrator registers a user, they specify the user name, password, expiration date, and other default options. The registration process creates an ID for the user or server and places it in the Domain Directory or in a file which must be given to the user to reside on the user’s workstation.

4.3.8 User ID, server ID, and certifier ID

Essentially Notes ID stores certificates and encryption keys. Three different types of Notes ID exist:

- User ID: The Notes ID for a Lotus Notes user.
- Server ID: The Notes ID for a Lotus Domino server.
- Certifier ID: This ID is treated differently because it is only used for certifying other Notes IDs. Therefore, it is much more important than other IDs for the organization. The certifier ID should be stored on a floppy disk in a safe place. It should not reside on a server hard disk where unauthorized users may be able to access it.

4.3.9 Contents of a Notes ID

When an administrator attempts to register a new user or server, the Lotus Domino Administrator workstation generates two RSA key pairs for that entity. One, 512-bit key length pair, is used for data encryption in non-North American countries. Another, 630-bit key length pair, is used for data encryption within the U.S. and Canada and for signature and authentication worldwide. The Domino Administrator then builds a certificate using the certifier’s private key to sign the certificate. The signed certificate is then placed in the Notes ID file.

After the registration process, the ID file contains:

- The user’s name and Notes license number
- Two public and private key pairs
- Two certificates for the user
- A certificate for each ancestor certifier

After the registration, encryption keys distributed by application developers may be added to allow encryption and decryption of fields in documents. The private key and the encryption keys in the ID file are encrypted using a key computed from the user’s password, so that only the owner can access it. Public information, such as the user’s name and public key, are not encrypted. Again using the organization shown in the previous section as an example, Figure 35 on

page 78 shows the ID files for user Jennifer in the East organization and for a server in the West organization.

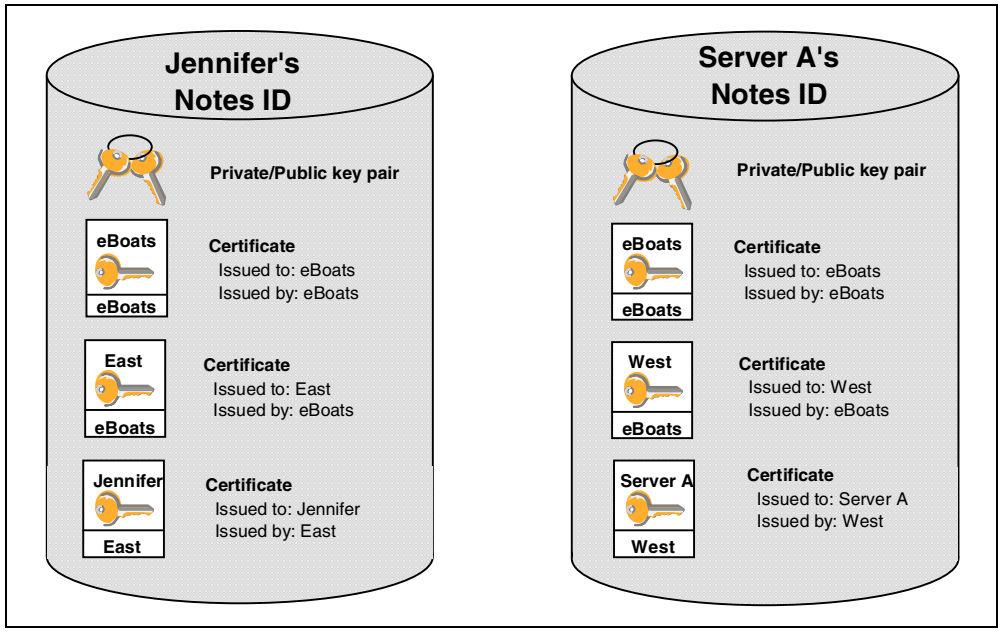


Figure 35. The content of Notes ID for the user and the server

4.4 Notes authentication

This section discusses how Lotus Notes and Domino authenticate each other in a session using Notes Remote Procedure Calls (NRPC). As we described already, Notes authentication is based on certificates, and it uses some public key-based cryptographic techniques.

Please note that this authentication technique is similar to the one specified by X.509. Refer to the following section for details on X.509-related techniques.

4.4.1 Validation and authentication

There are two phases in verifying a user's or server's identity in Notes. The first phase, called *validation*, is the process of reliably determining the sender's public key. In other words, the validation is the preparation phase for the actual authentication.

Notes uses the following rules when deciding to trust a public key:

- Trust the public key of any of your ancestors in the hierarchical name tree because they are stored in your ID file.
- Trust any public key obtained from a valid certificate issued by any of your ancestors in the hierarchical name tree.
- Trust any public key certified by any trusted certifier and belonging to one of the certifier's descendants.

4.4.1.1 Phase 1: Validation

Let us now see how these rules are applied in the validation process. The user ID file for Jennifer contains everything she needs to identify herself and establish

her credentials. When she requests a session with a server, the first step is to send to the server all of the certificates from the ID file (both the user's own certificate and the chain of certifiers' certificates that support it). Figure 36 illustrates the validation process that follows.

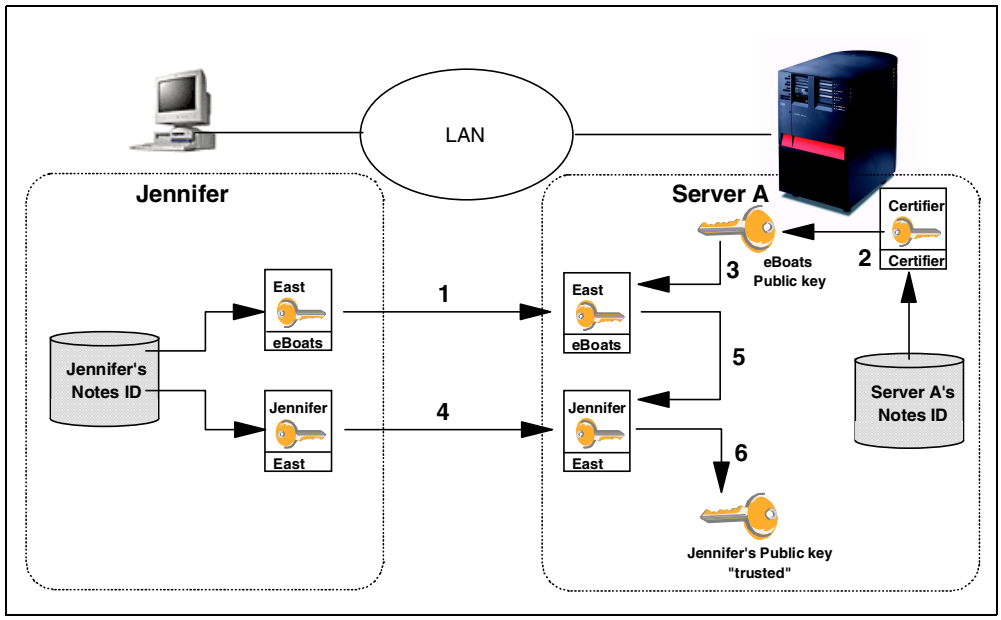


Figure 36. The validation process

The numbered steps shown in Figure 36 are described as follows:

1. Server A reads the East certificate that Jennifer sent from its ID file. This was signed by eBoats. ServerA is interested in it because East is the certifier of Jennifer's certificate.
2. Server A reads the eBoats public key from its own ID file. According to rule 1, Server A will trust the public key of any ancestor that is stored in its ID file.
3. Server A uses the public key of eBoats (which is trusted because it is in the server's ID file) to verify that the certificate of East/eBoats is valid. According to rule 2, if you trust the public key of the ancestor, you will trust any public key obtained from certificates issued by the ancestor.
4. Server A reads the certificate that was sent from Jennifer's ID file. This was signed by East.
5. Server A uses the public key of East/eBoats, which now is trusted, to verify that the Jennifer Minge/East/eBoats certificate is valid. According to rule 3, trust any public key certified by any trusted certifier and belonging to one of the certifier's descendants.
6. Server A has now reliably learned Jennifer's public key.

The same process is followed in reverse so that Jennifer can reliably learn Server A's public key.

4.4.1.2 Phase 2: Authentication

After the validation process finishes, the authentication process begins.

Authentication is a proof of identity. The validation process described in the presentation has not completely proved who each of the session partners is, because all that has been presented so far is certificates. A certificate associates the user with a public key and tells the recipient that the public key can be trusted. However, to prove that users really are who they claim to be, they must show that they hold the private key that matches the public key in the certificate. The authentication process achieves this with a challenge/response dialog between a workstation and a server, or between two servers when either is running database replication or mail routing.

To continue the previous example of Jennifer accessing Server A, see Figure 37. The following procedure shows a simplification of the actual process, and is intended to illustrate what happens in a manner that's easy to understand. The numbers shown in Figure 37 are further explained in the list that follows.

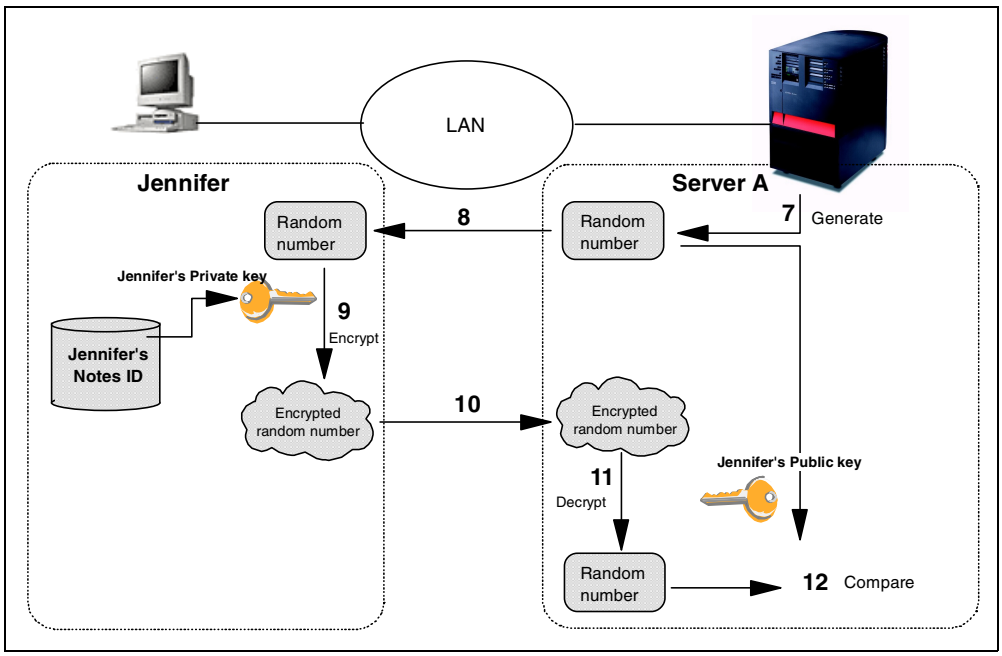


Figure 37. The authentication process

1. Server A generates a random number and a session key, and encrypts both with Jennifer's public key.
2. Server A sends the encrypted random number to Jennifer.
3. Jennifer receives the challenge and decrypts it with her private key.
4. Jennifer sends back the decrypted number to Server A.
5. Server A compares Jennifer's response to the original random number.
6. If the result is the same as the original random number, Server A can trust that Jennifer really is who she claims to be.

As with validation, authentication is a two-way procedure. Jennifer now authenticates Server A using the same challenge and response process in reverse.

The actual algorithm is complex but efficient. It avoids any RSA operations on subsequent authentications between the same client-server pair. It also

establishes a session key that can be used to optimally encrypt the messages that follow authentication.

4.4.2 Switching off certificate-based authentication

You can give Notes users anonymous access to a Domino server or Domino servers anonymous access to another Domino server. This lets users and servers access the server without that server authenticating them. This is most useful for providing the general public access to servers for which they are not cross-certified.

To allow anonymous access, open the Server document in the Domino Directory. Select the **Security** tab, and modify “Allow anonymous Notes connection” of the security settings as appropriate. See Figure 38.

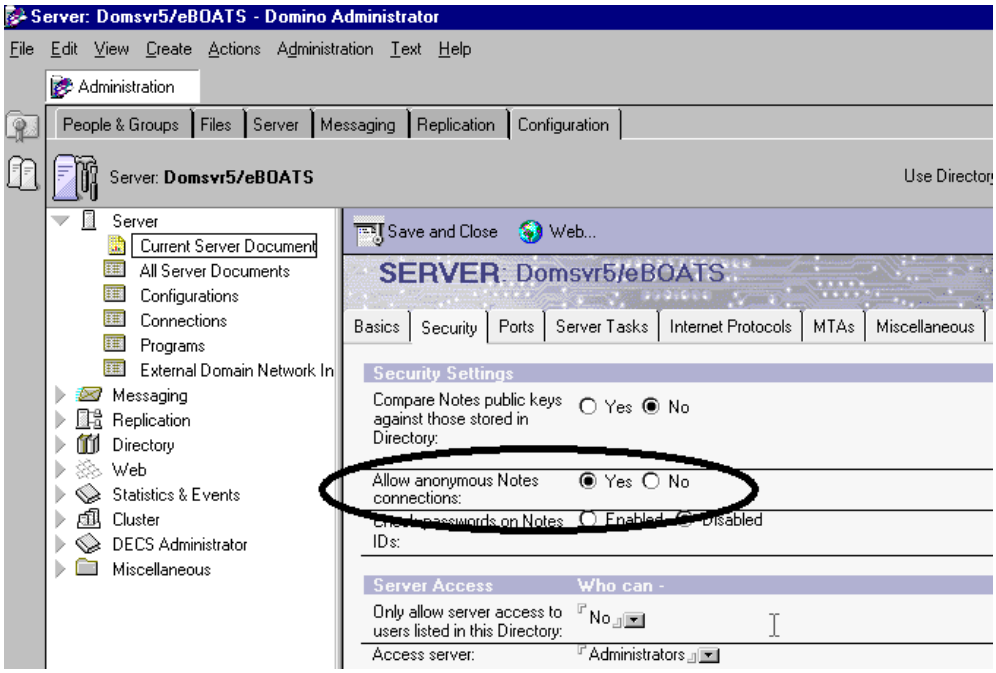


Figure 38. Setting for allowing anonymous Notes connections

When you set up anonymous server access, Domino does not record the name of the person or server from the ID file, for example, in the Log file or in the User Activity dialog box.

If you are in a hierarchical certification environment and attempt to connect to a server which is set for anonymous access and the server can’t authenticate you, you see the following message in the status bar:

```
Server X cannot authenticate you because: the server's Address Book does not
contain any cross-certificates capable of authenticating you. You are now
accessing that server anonymously.
```

4.5 Web browser client authentication and validation

Today’s Internet security is based on public-key X.509 certificate technology. However, as mentioned previously, Lotus Notes has offered certificate-based security for many years.

In Notes and Domino R5.0, there is an integrated infrastructure for the existing native Notes-certificates and the Internet X.509-based certificates.

4.5.1 X.509 certificate

Although there have been several proposed formats for public key certificates, most commercial certificates available today are based on the international standard ITU-T Recommendation X.509 (formerly CCITTX.509).

X.509 certificates are used in such secure Internet protocols as:

- Secure Socket Layer (SSL)
- Secure Multi-purpose Internet Mail Extension (S/MIME)
- Secure Electronic Transaction (SET)

4.5.2 X.509 standard

Originally, the X.509 standard was intended to specify the authentication service for X.500 directories. Directory authentication in X.509 can be done using either secret-key techniques or public-key techniques. The latter is based on public-key certificates. At present, the public-key certificate format defined in X.509 standard is widely used and supported by a number of protocols in the Internet world. X.509 standard does not specify a particular cryptographic algorithm. However, apparently RSA algorithm is the most broadly used one.

The initial version of X.509 was published in 1988. The public-key certificate format defined in this standard is called X.509 Version 1 (X.509v1). When X.500 was revised in 1993, two more fields were added, resulting in the X.509 Version 2 (X.509v2) format. X.509 Version 3 (X.509v3) was proposed in 1994. X.509v3 extends v2 to address some of the security concerns and limited flexibility that were issues in Versions 1 and 2. The major difference between Versions 2 and 3 is the addition of the extensions field. This field grants more flexibility because it can convey additional information beyond just the key and name binding. In June 1996, standardization of the basic v3 format was completed.

4.5.3 X.509 certificate content

An X.509 certificate consists of the following fields:

- Version of certificate format
- Certificate serial number
- Digital signature algorithm identifier (for issuer's digital signature)
- Issuer name (that is, the name of the Certification Authority)
- Validity period
- Subject (that is, user or server) name
- Subject public-key information: Algorithm identifier and public-key value
- Issuer unique identifier: Version 2 and 3 only (added by Version 2)
- Subject unique identifier: Version 2 and 3 only (added by Version 2)
- Extensions: Version 3 only (added by Version 3)
- Digital signature by issuer on the above fields

Standard extensions include subject and issuer attributes, certification policy information, and key usage restrictions, among others. After this introduction to X.509 certificates, we look at how they are used by Notes and Domino R5.0, in relation to SSL.

4.5.4 Secure Socket Layer (SSL)

The SSL protocol was originally created by Netscape Inc., and is now widely implemented in most Internet-based client/server software. SSL uses a number of cryptographic techniques, such as public key and symmetric key encryption, digital signatures, and public key certificates. SSL Version 3.0, the current version, is a security protocol that:

- Encrypts information sent over the network from client and server (confidentiality)
- Validates that the message sent to a recipient was not tampered with (data integrity)
- Authenticates the server, using RSA public key methods (authentication)
- Authenticates the client identity (authentication, new in Version 3.0)

SSL Version 3.0 also supports the X.509 v3 certificate format, new key-exchange and encryption algorithms, and improved server performance through caching.

Normally SSL is considered as a way of securing HTTP traffic. However, SSL has the advantage of being application protocol independent, which means that it can run on top of any TCP/IP application protocol, such as NNTP, POP3, and LDAP. SSL connections use a different port number for each application protocol that it is encrypting. For example, when running SSL over HTTP, port 443 is used, as assigned by the Internet Assigned Numbers Authority.

4.5.5 How SSL operates

There are two parts to SSL:

- **The handshake**: Session partners introduce themselves and negotiate session characteristics.
- **The record protocol**: Session data is exchanged in an encrypted form.

4.5.5.1 The SSL handshake

Before data can be sent or received over a connection protected by the SSL protocol, a session must be established. SSL requires a certain method of establishing a secured connection. This method of exchanging specific messages prior to the user session is called the *SSL handshake*. Digital certificates play an important role within the SSL handshake, which has to flow in a predefined order using standard formats.

The handshake protocol is responsible for negotiating a *Cipher Spec* and generating a shared secret key. The Cipher Spec defines the kind of encryption (for example, DES, RC, RC4) and authentication (for example, MD5, SHA) algorithms that can be used for the communication session. Currently there are two versions, Version 2.0 and Version 3.0, of the SSL protocol available, but only Version 3.0 supports client authentication.

Figure 39 on page 84 shows the flow of the SSL handshake using a server certificate only.

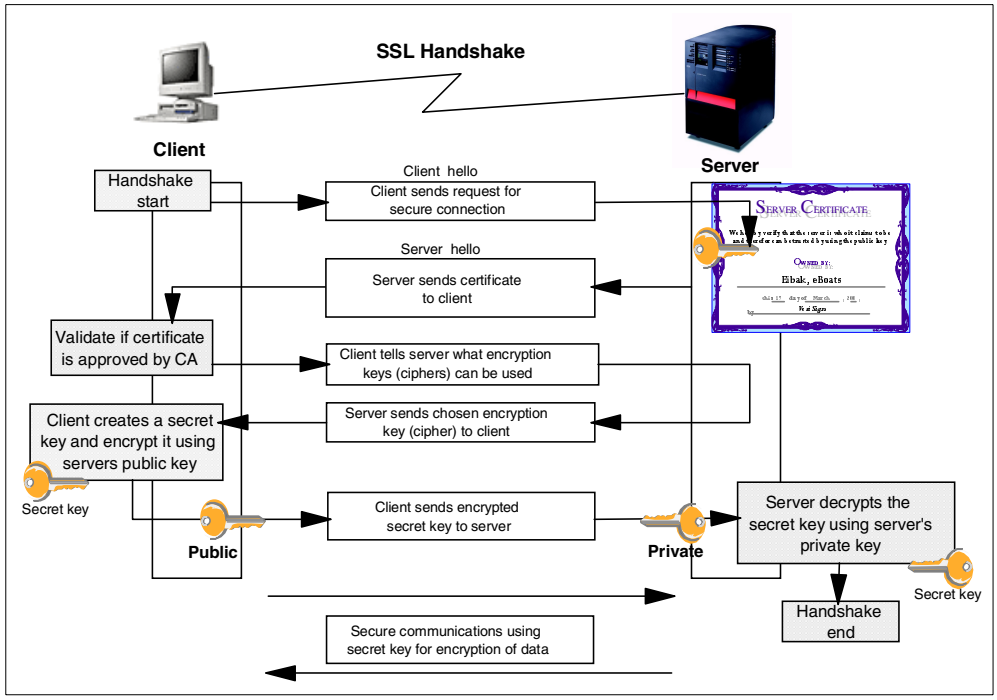


Figure 39. SSL handshake with server certificate only

The process shown in Figure 39 is explained here:

1. The client sends a request to connect to an SSL-enabled server. This is done by sending the request to a specific port used for secure connections. Normally this is the well-known port number for the secure version of the protocol being used. When using HTTP, the request is initiated through the URL starting with https. The request is also called the Client Hello message. This message is also used to exchange attributes, such as protocol version, session ID, and so on.
2. The server sends back its Server Hello message and its certificate.
3. The client checks if the certificate was issued by a CA it trusts. If so, it checks that the certificate is valid. If either of these checks fails, the client can cancel the connection or choose to proceed without authentication.
4. The client tells the server what ciphers, or types of encryption keys, it can use for communication.
5. The server chooses the strongest common cipher and informs the client about its selection.
6. Using that cipher, the client generates a session key (an encryption key to be used only for this session) and encrypts it using the server's public key.
7. The client sends the encrypted session key to the server.
8. The server receives the session key and decrypts it using its private key.
9. This completes the handshake. The client and server use the session key to encrypt and decrypt the data they send and receive.

4.5.5.2 SSL record protocol

The SSL record protocol specifies a format for these messages. In general, they include a message digest, using the MD5 algorithm, to ensure that they have not been altered and the whole message is then encrypted using a symmetric cipher. Usually, this uses the RC2 or RC4 algorithm, although DES, Triple-DES and IDEA are also supported by the specification.

For more information about the SSL protocol and the SSL handshake, refer to the SSL 3.0 Internet draft at: `http://www.netscape.com/eng/ssl3/index.html`

You can see from this example that there is significant additional overhead in starting up an SSL-session compared with a normal HTTP connection. The protocol avoids some of this overhead since the negotiation and authentication are done only once at the beginning of the session.

Note

The US National Security Agency (NSA), a department of the United States federal government, imposes restrictions on the size of the encryption key that may be used in software exported outside the U.S. Beginning about July 1997, changes were made to these rules, where the U.S. Government now allows the export of 128-bit encryption keys for any International banks and US companies with foreign offices. Products that support Server Gated Cryptography (using strong encryption) are still not allowed to be exported to embargoed countries. VeriSign currently is authorized to issue these special certificates called VeriSign Global Secure Server ID.

The RC2 and RC4 algorithms can achieve this by using a key in which all but 40 bits are set to a fixed value. International export versions of software products have this hobbled security built into them. SSL caters to mismatches between the export and non-export versions in the negotiation phase of the handshake. For example, if a US browser tries to connect with SSL to an export server, they will agree on export-strength encryption.

On 15 January 2000, the U.S. Government relaxed their restrictions on the worldwide shipment of stronger encryption (defined as 128-bit support). The implication to Lotus, therefore, is that worldwide shipment of 128-bit encryption is now permitted. Customers will no longer be required to order and choose between three different kits (North American, International English, and International English for France). The North American version is available to individual customers and commercially worldwide (with some restrictions).

There will be a new edition with stronger encryption that will be translated for shipment worldwide. The changes will only be implemented in a post 5.0.3 release.

4.5.6 SSL deployment considerations

In practice, there are a few questions that you need to have answers to before deploying security on your site. For example, will you be using server authentication or client authentication? Who are your users and from where are they connecting? Are you in a private network or intranet, or will your server be on the Internet? Do your browser users trust you? And, more importantly, do you

trust the people that are connecting to your server? These kinds of questions lead us into four broad areas of discussion:

- Server authentication
- Client authentication
- External Certificate Authorities
- Internal Certificate Authorities

4.5.6.1 Server authentication

If the browser user that connects to your server has a trusted certificate or root certificate common with your server, this provides the Web browser user with confidence that your server is what you say it is. If you want to control what the user can see on your secure site, you still need to manage user names and passwords for each user.

From the point of view of a browser user connecting to a Web site using SSL, the whole negotiation and authentication process can be quite transparent. To establish an SSL connection, the URL prefix must change from http:// to https://. Once the SSL connection is established, the browser gives the user a visual indication. In the case of Netscape Communicator, this is a closed padlock symbol at the lower left of the display as shown in Figure 40.

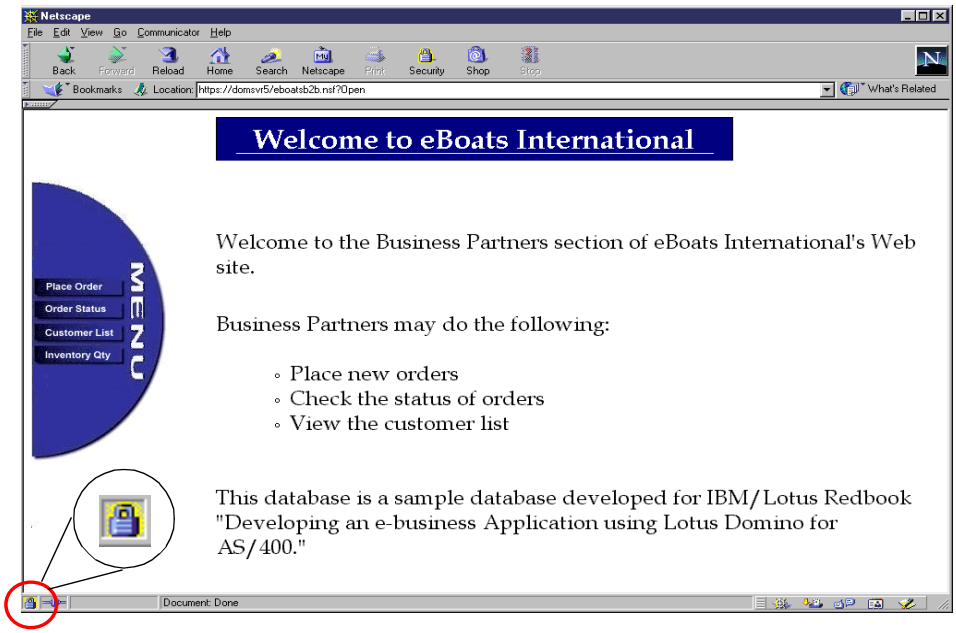


Figure 40. Example on a secure connection authenticated by the server

4.5.6.2 Client authentication

With client authentication, we take authentication one step further. You as the server will want to know if you can trust the Web browser user. The Web user exchanges a client certificate, which is signed by a Certificate Authority (CA) that is, a third party that you trust, or it may even be you (see 4.5.6.4, “Internal Certificate Authority” on page 88). This provides you with confidence that the browser user represented by the certificate is the person you expect. You also have the added advantage of not having to manage passwords for these users, which is one less administrative burden, since their authenticity is now vouched for by the CA.

From the point of view of the Webmaster, SSL is also quite simple. First, the Webmaster needs to generate a key pair for the server and obtain a certificate for it. Normally this involves providing documentation to a certifying authority and paying an annual fee, although it is also possible to generate your own certificates for testing and intranet use. It is also possible to be your own Certificate Authority within your own organization. The difference between using an Internal CA or a third-party CA, is basically a matter of trust. Within an intranet organization, you may decide that it is reasonable for your employees to trust any server within the intranet, by the very nature that it is internal to your organization.

4.5.6.3 External Certificate Authority

If you are deploying your Web server onto the Internet, there is the issue of how and why a Web user on the Internet can believe you are who you say you are. This is particularly pertinent if they have to send credit details to you. Using an external CA (that is, a trusted third party) that your browser trusts solves this problem.

As shown in Figure 39 on page 84, authentication in SSL depends on the client being able to trust the server's public key certificate. A certificate links the description of the owner of a key pair to the public part of the key. The validity of a certificate is guaranteed by the fact that it is signed by some trusted third party, the Certificate Authority (CA). But how does a certifying authority become trusted? In the case of an SSL-capable browser, the certificates of trusted authorities are kept in a key database, sometimes called a key ring file.

The list of top-level authorities contains a set of certificates known as *root certificates*, which are pre-installed when you get the browser. Figure 41 on page 88 shows part of the list of CA certificates provided by Netscape Communicator.

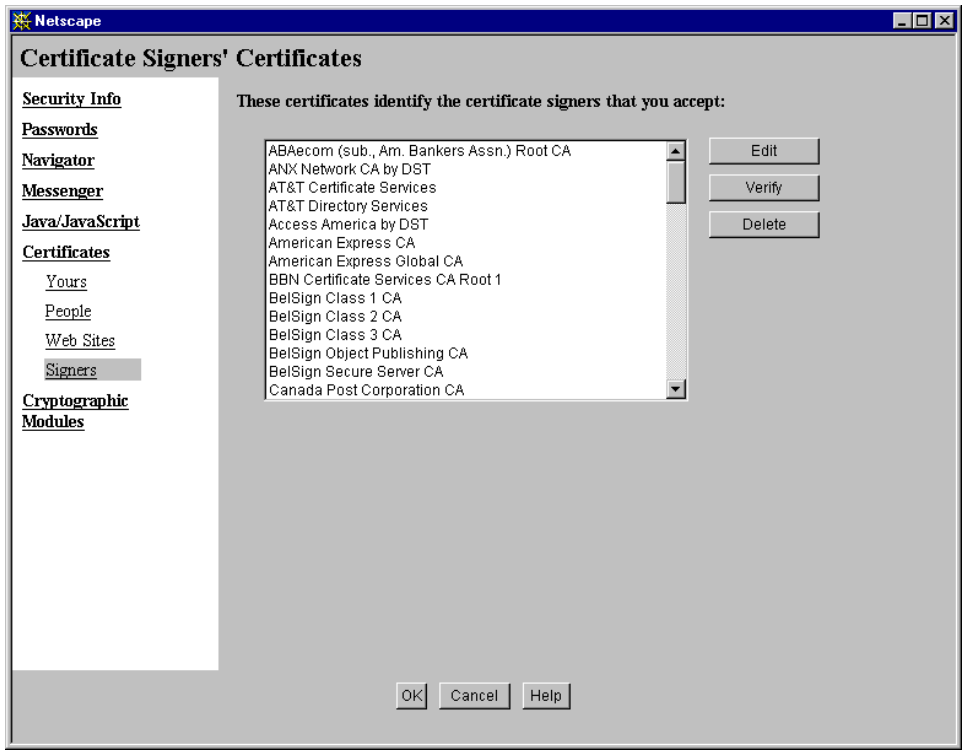


Figure 41. Certificate Signer's Certificates

This approach has the benefit of being very simple to set up. A browser can authenticate any server that obtains a public key certificate from one of the CAs in the list, without any configuration or communication with the CA required. However, there are some problems arising from this method. The first is that a new CA will not automatically be recognized until the browser (wherever it may be) has been updated. The second problem is that there is no way for certificate revocations to be processed. For example, if a CA determines that a public key owner is fraudulent after a certificate is issued, the certificate will remain usable until it expires, without the end user being aware of any concern.

The browser vendors have a two-part scheme to overcome the first problem (new CAs):

1. There is a special MIME format, application/x-x509-ca-cert, which allows a browser to receive a new CA certificate that has been signed by one of the known CAs. This format is specified in PKCS #7 and is available at:
   `http://www.rsa.com/rsalabs/pubs/PKCS/index.html`
2. The browser will tell you that you are connecting to a secure server whose certificate is not from a known CA. You can then elect to trust just that server (that is, not the CA that signed the server's certificate).

For Internet applications, you should purchase a certificate from one of the known CAs, such as VeriSign.

4.5.6.4 Internal Certificate Authority

There are instances when you may consider becoming an Internal Certificate Authority. This very much depends on who your browser users are. If they are internal users, you are dealing with a smaller risk and so it may be acceptable for

you to be the CA, since your Web users will trust you. It also means that you will not need to pay a yearly charge to the external CA.

As mentioned earlier, the browser comes pre-installed with a list of top-level authorities. This leaves you as the CA administrator with two options: either to install your new CA certificate in all your browsers, as described earlier, or prepare your users for the message that they will connect to a site that has been signed by a CA that the browser does not recognize.

Figure 42 and Figure 43 on page 90 show how Netscape Communicator warns you about a site that has a certificate from an unknown CA and then allows you to trust the site anyway.

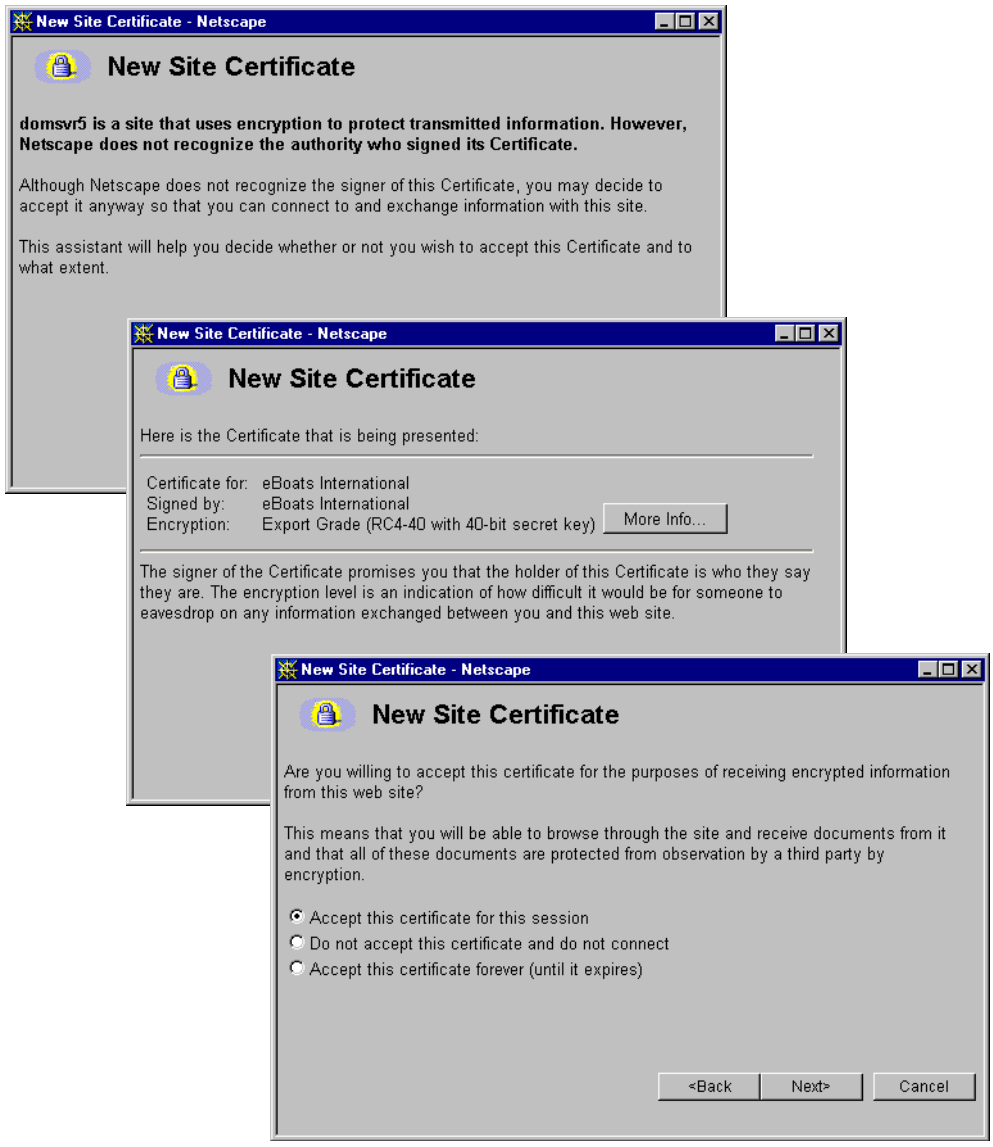


Figure 42. Netscape warning when receiving a certificate from an unknown CA (Part 1 of 2)

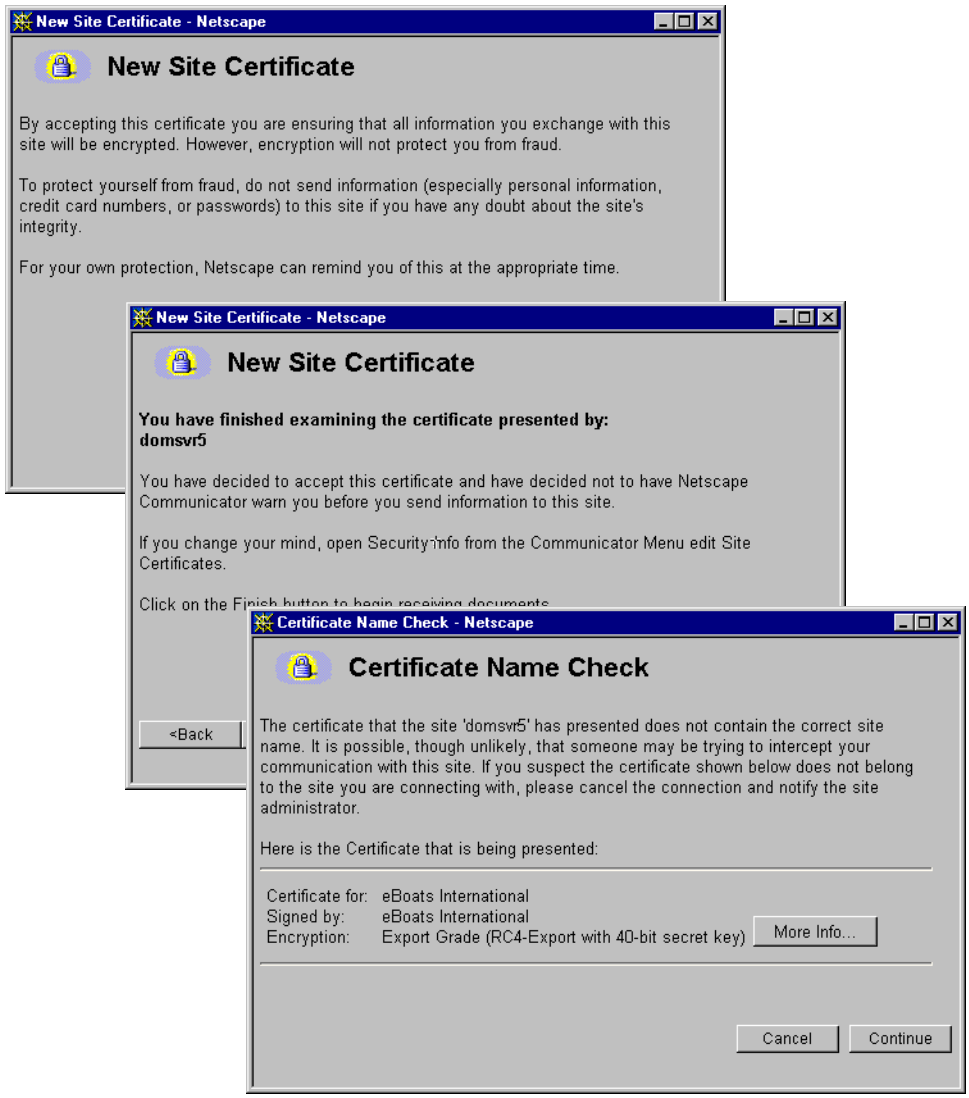


Figure 43. Netscape warning when receiving a certificate from an unknown CA (Part 2 of 2)

Note

This does not make the CA trusted. You see the same warnings if you access a second server with a certificate signed by the same CA.

4.5.7 Comparison between Notes security and SSL

In this chapter, we have described many security functions that need to be implemented. Table 10 attempts to summarize some of these, comparing security functions at the application level in the Notes and Web environments.

Table 10. Comparisons between Notes security and SSL

Function	Notes implementation	Web + SSL implementation
Access Control	Multi-level/role-based access control, based on user authenticated Common Name. Full access control over what the client can see and do at the field, section, document, form, view, database, and server level.	Exclusive access to specific HTML files, directories, executables: that is, view or not view, execute or not execute. Access control methods based on X.509 client authentication not yet developed.
User Management	Distributed directory of user information (Domino Directory) includes personal details and Notes and X.509 certificates. Ability to group users using distributed group profiles. LDAP allows for cross directory access and management.	Basic authentication: user ID and password file maintained separately on each server. Ability to create simple group of users. SSL Client authentication: Directory of user information includes personal details and public-key certificate information. LDAP allows for cross-directory access.
Certification scheme	Public-key Certificate Authority function provided by domain and sub domain servers. Integrated into the Domino directory. Cross-certification across multiple domains, in a controlled fashion. Manual registration by domain administrator.	Current X.509v3 certification: Small number of well-known certification authorities allow server to be recognized. Ability to become a Certificate Authority and distribute to browsers. LDAP allows cross-directory access.
Authentication	RSA public-key technology used to digitally sign a challenge string. Both Client and Server authentication.	RSA public-key technology used to digitally sign a challenge string. Both Client and Server authentication by SSLv3.

Function	Notes implementation	Web + SSL implementation
Encryption methods	RSA technology (630/512-bit) used to encrypt key material that is subsequently used in a symmetric cipher for bulk data encryption. Encryption integrated with access controls. Bulk data encryption uses RC2/64 bit with 24 bits, accessible by the US intelligence agencies in non-US version of Notes. methods	SSL: RSA technology (key length varies) used to encrypt key material that is subsequently used for bulk data encryption. RC2/4 or other technologies for bulk encryption. 128-bit key usually used for US and some categories of companies (such as banks) for non-US, and 40 bit for others.

4.5.7.1 Enabling SSL at a database level

To protect transactions in individual databases, for example in databases used for commercial transactions, it is possible to enable SSL at the database level. To implement this, open the property box for the database, and select **Web Access: Require SSL Connection**. To open SSL-protected databases, Web users must include the SSL certificate in their browser configuration.

Note that the server administrator actually has three options in allowing access over the SSL port:

- **Anonymous access over the SSL port**: Web users can connect to the server using the SSL port. However, the server allows anonymous users.
- **Name and password access**: Web users connect to the server over the SSL port and authenticate using name and password.
- **Access through client certificates**: Web users have been issued an X509 client certificate and connect to the server over the SSL port. They are authenticated using this client certificate.

Depending on which of these types of access the server allows for the SSL port and what the database ACL allows, the user may access this database anonymously, may be authenticated with name and password authentication, or may use a Client certificate.

4.5.7.2 Enabling SSL at a servlet level

There is no server setting for forcing SSL on servlets. One possibility is to have the servlet itself force a redirection if the connection is not using SSL. More details can be found in 11.4, “Enabling SSL at a servlet level” on page 302.

4.6 Considerations for the AS/400 Digital Certificate Manager

There may already be a digital certificate infrastructure implemented in your AS/400 environment, using the AS/400 Digital Certificate Manager (DCM).

On the AS/400 system, there are many TCP/IP server applications available that can communicate over SSL, such as:

- IBM HTTP Server for AS/400
- IBM AS/400 Client Access Express
- Management Central
- Lightweight Directory Access Protocol (LDAP)
- IBM eNetwork Personal Communications
- SecureWay Host On-Demand
- Java applications using the AS/400 Toolbox for Java
- Distributed Relational Database Architecture (DRDA)
- Distributed Data Management (DDM)

User applications can also communicate over SSL. To use and manage digital certificates in user applications, the AS/400 system provides a set of application programming interfaces (APIs) that can be used, for example, with RPG. The AS/400 system also provides the support for digital certificates in Java programs. Whether you are using the IBM WebSphere Application Server to run Java servlets, using the AS/400 Toolbox for Java, or writing Java applications that are communicating over sockets, the AS/400 system supports classes to use digital certificates for this kind of application.

The AS/400 Digital Certificate Manager (DCM), is an application for managing digital certificates. It includes the ability to create and store certificates on the AS/400 system, to validate certificates, to associate certificates with user profiles, and applications. Information about the AS/400 Digital Certificate Manager (DCM) can be found in the redbook *AS/400 Internet Security: Developing a Digital Certificate Infrastructure*, SG24-5659.

The AS/400 DCM can't be used as a CA for Domino for AS/400.

Note the following points taken from the Lotus Domino Administrator's help:

1. The Domino administrator has to request a server certificate from a CA. Requesting a server certificate involves creating a public/private key pair and a certificate request file. Usually, the request data is displayed on the screen in a base64-encoded ASCII text form. The administrator then copies and pastes the data into a certificate request Web page of a CA and sends the request.
2. The CA receives the data, verifies the correctness, and signs the certificate. Once the CA signs the certificate, the certificate is sent back to the requestor.
3. The Domino administrator receives the signed certificate and merges it into the certificate store (key ring file). To use the server certificate, it must be received or merged on the system on which the request was created. Normally, the applications take care this and refuse to receive a certificate for which no request was created. Remember, the key pairs and the certificate build one unit and they belong together to be used, for example for SSL server support.

The process of receiving server certificate requests and signing the certificates is not implemented in the AS/400 DCM. This is the reason why the AS/400 DCM can't be used to issue server certificates in the way it is explained in the Domino documentation.

Chapter 5. Development options

Once your Domino server is set up, you can begin designing your application elements for a system to be used over the Web.

One of the key requirements is for users of Notes applications to access data or transactions in other enterprise systems. There are several different ways of achieving this, and these are shown in 5.4, “Tools for enterprise integration” on page 117.

There are two different client systems:

- Web browser
- Notes client

The Web browser implementation is different because the Lotus Domino server provides a Web server function, and does the conversion between HTTP protocol and the Notes environment. The Web browser has fewer functions than the Notes client.

This chapter focuses on the development options to design your e-business applications.

5.1 Templates

A template is a skeleton that contains design elements, but no documents. When you use a template to create a database, the database receives the design elements from the template. You can use the Designer templates as is or customize them. Or, you can create a template by creating a database with the Notes Template File (NTF) file extension and the “Database is a template” property. This property enables the master template to distribute design changes automatically to databases created from it. Databases that inherit their designs from master templates receive the latest changes through a nightly server task.

If, for some reason, you do not want a database to automatically inherit designs through the nightly server task, you have two options:

- Do not put the template on a server that runs the nightly update task.
- Do not set the “Database is a template” property.

You can then distribute changes to the database design by using the Replace Design command, specifying the server where the template is located and selecting the template. Make sure to set the database properties so that the database does not automatically inherit the design through the nightly server task.

Using a template, you can establish design standards for use throughout your company. In large companies, a central development group usually designs and manages templates to provide consistent designs and speed up distribution of new databases. Use a template to standardize similar types of applications, for example, all discussion databases, or to store individual design elements, such as fields, forms, views, folders, navigators, and agents that you can use in a variety of applications.

5.1.1 Customizing the template

To customize a Designer template, select **File->Database->New Copy** to copy the original template and inherit the original design. Give it a different file name in the Copy Database dialog box to prevent future releases from writing over your customized template.

Here's a list of changes you can make when you customize a Designer template:

- Leave the original Designer template properties alone so that existing databases that inherit their design will continue to be synchronized to the template.
- Change the newly copied template file name in the Copy Database dialog box (**File->Database->New Copy**) to a name that indicates its intended use.
- Change the newly copied template name (**File->Database->Properties**) to a new name, indicative of the new template purpose.

If existing databases that inherited the original Designer template design need to inherit the design from the newly created template, edit the database properties of those databases to reflect the name of the newly copied template.

5.1.2 Advantages of standardizing with templates

The advantages of using templates in Notes development are:

- Anyone can quickly and easily create a new database.
- Users only need to know the File->Database->New command. They never have to modify the design and don't need to know anything about database design.
- Developers and experienced users can save design time.
- Forms, views, and agents copied from a template require no additional design work or updates. Using a pre-designed form or view that contains complex formulas or a large keywords list reduces the chance of design errors and requires less testing time before a database rollout.
- Databases appear consistent to users.
- View, forms, and fields associated with a template use the same names in all databases. This allows users to apply their knowledge of one database to many databases.

Note

The template does not usually control the database icon, the About This Database document, and the Using This Database document. If you want an icon, About document, or Using document to inherit all design changes, go into Designer. Choose **Resources - Other**, select the icon or document, and make changes in the Properties box so that changes are inherited from the template. The default is not inherited. You can also change these design elements by manually copying and pasting the redesigned elements into databases linked to the template.

In addition, do not create private agents or folders in templates. Their changes can't be distributed automatically. If a private agent or folder exists in a template, you receive error messages when the design is refreshed or replaced. In the case of a shared or private-on-first-use folder, if the folder in the template has been opened or tested, a private folder exists and that folder generates errors when you try to refresh a design.

5.1.3 Domino Designer templates

When you install Domino Designer, you can install application templates. Some of them allow to instantly create applications that can serve Web browser users as well as Lotus Notes users:

- Discussion - Notes & Web (R5.0): To create an electronic conference room
- Document Library - Notes & Web (R5.0): To create a document storage application
- TeamRoom (R5.0): For team collaborative operations
- Site Registration (R5.0): To manage Web users registration

For an in-depth look at features of these templates, see the Domino Designer Templates Guide, available on the Notes.Net Web site at: `http://www.notes.net`

5.2 Design elements

This section describes how to create and manage Domino databases. This chapter also includes a glossary of Domino design terms that an application developer needs to understand when creating a Domino application.

5.2.1 Domino databases

The term “Domino database” refers to both Domino and Web databases. What makes it a Web database is the viewing mechanism (a Web browser instead of a Notes client) and the fact that it resides on a Domino server running the HTTP server task.

Traditional Web sites consist of different kinds of pages and the associated compound elements that are organized in hierarchical directory structures. When an HTTP request is issued to display a page, a new HTML coded file is opened.

With Domino, the Web site is structured through Domino databases designed in the Notes object store format. When an HTTP request is issued to display a page, a Notes element is opened through a Universal Resource Locator (URL) command, and Domino translates it for viewing as a Web page.

5.2.2 Forms and fields

The form is the skeleton provided to users to enable them to enter data, either by typing or using buttons. There is usually at least one form in a database, although a typical business application has many forms, each targeted to the type of information that the user wants to save in the database.

The form contains all the design elements: fields to store the user's information, static text, buttons, sections, images, and subforms that help the user to enter the data into the database.

5.2.2.1 Form elements

Form elements are the components you use to create a form's aesthetics and function. Table 11 lists the elements options in a Form.

Table 11. Form elements

Elements	Description
Fields	Fields are the design element that collect data. You can create fields only on forms. Each field on a form stores a single type of information. A field's *field type* defines the kind of information a field accepts. You can place fields anywhere on a form.
Subforms	A subform is a collection of form elements stored as a single object. A subform can be a permanent part of a form or can appear conditionally, depending on the result of a formula. Subforms save redesign time. When you change a field on a subform, every form that uses the subform changes. Common uses of subforms include adding a company logo to business documents or adding mailing label information to mail and memo forms.
Access-controlled sections	You can restrict access to a section by creating an access control list.
Layout region	A layout region on a form or subform is a fixed-length design area in which related elements can be dragged and moved easily and can be displayed in ways not possible on regular forms and subforms. A layout region can contain static text, graphics, buttons, and all fields except rich text fields. You can hide or collapse a layout region and all its components under certain conditions. Layout regions are not supported for Web applications.

Elements	Description
Embedded elements	Embedded elements, such as the file upload and the group scheduler that require programming support, can be used on a form only.
Text	Text is often used to label fields so users understand the purpose of each field. For more information on formatting text, see Notes 5 help.
Horizontal rules	Add horizontal rules to separate different parts of a form or to make a form more interesting visually.
Tables	Use tables for summarizing information and aligning elements such as fields and graphics in rows and column. A table placed on a form appears in every document created with the form.
Sections	A section is a collapsible and expandable area defined on a form. It can include fields, objects, layout regions, and text. An access-controlled section allows only certain users to edit the fields in the section.
Graphics	When you place a graphic anywhere on a form, it appears on every document created with the form. For example, on a form for correspondence, placing your company logo at the top of the form creates a letterhead.
Imagemaps	Imagemaps are graphics you enhance with programmable hotspots that perform some action when clicked by a user. Imagemaps are often used as navigational structures in an application.
Links	Within a form, you can add links to databases, views, specific documents, or URL links that open pages on the Internet.
Automation	Form actions, buttons, or hotspots automate simple or complex tasks.
Applets	Use Java applets to include small programs, such as an animated logo or a self-contained application, in a form.
Attachments	You can attach files to a form so users can detach files locally or launch them from every document created with the form.
Embedded elements	Embed the following elements in a form: an outline, view, folder pane, import navigator, navigator, and date picker. Use these elements alone or combine them to control how users navigate through your application.

Elements	Description
HTML	If you have existing HTML or you prefer HTML, you can use it on forms. Use existing HTML by importing or pasting it into your form, or you can write your own.

5.2.2.2 Forms and documents

When a user creates and fills out the information in a form and saves it, the information is saved as a document. When a user opens the document, the document uses the form as a template to provide the structure for displaying the data. When designing forms, you should consider where and how the resulting documents will be displayed.

A form is stored in the database it was created in and used to display all associated documents. However, there may be times when you are mailing a document to a database that does not have the form that was used to create the document. In these cases, you can designate the form to be stored with each document created from it. Storing the form with each document consumes more memory.

When a user opens a document, Domino uses the rules in Table 12 to determine which form to use to display it.

Table 12. Forms options

Condition	Form used to display document
If the form used to create the document is available and there is no form stored in the document and no form formula.	The form that was used to create the document. The original form name is stored in a hidden field called “Form” in the document. To find the value of the field, you can check the Document Properties box under the Fields tab.
If a form is stored with the document	The form stored with the document. When a form is stored in a document, the form name is stored in an internal field called $Title.
If the view has a form formula.	The form is determined by the view's form formula.
If the form used to create the document is not available in the database.	The default form for the database. Each database can have only one default form, which is marked with an arrow in the Forms list.

5.2.2.3 Storing a form with each document

Storing the form with each document allows the document to display correctly even in a database where the form is missing, renamed, or deleted. This feature uses more system memory and may require as much as 20 times more disk space. It can also cause additional work if you change the form design because there is no easy way to update all of the stored copies of the form. In general, store a form in a document only under these conditions:

- The database to which documents are mailed or pasted does not contain a copy of the original form.

- The database to which documents are mailed or pasted doesn't share an alias with the original form.
- The form contains an embedded OLE object or a subscription, and you want documents to reflect any changes to the object.
- You selected “Include in Search Builder” in the Form Properties box and want the form's static text to be searchable.
- The documents created with this form are stored as encapsulated databases and mailed to cc:Mail users.

To store a form with each document, perform the following steps:

1. Open the form.
2. Click **Design->Form Properties**.
3. Click the **Form Info** tab.
4. Select **Store form in document**.
5. Switch to Database Properties in the drop-down list on the Properties box, and select **Allow use of stored forms in this database.**

Overriding the stored form

When a form is stored in a document, the form name is stored in a hidden field called $Title. Additional information is stored in the $Info, $WindowTitle, and $Body fields. To use a different form to display the document, create an agent that deletes this stored form information and designates another form to display the document.

Shared fields and documents with stored forms

If the form contains a shared field, that field is converted to a single-use field in the copy of the form that is actually stored in the document. This ensures that if a copy of the document is stored in a database that does not contain the shared field definition, the field can still be used. In the original form, the field is still defined as shared.

Form formulas

To override the default form selection, write a form formula for a particular view. For example, you can write a form formula that uses one form to display all fields when a user edits a document and a different form that re-sequences or omits fields when a user reads a document. Since form formulas apply only to a specific view, documents created in other views do not use the form formula.

To designate a default form for a database, perform the following steps:

1. Open the Form Properties box.
2. Click the **Form Info** tab.
3. Select **Default database form**.

Alternatives to storing forms

As an alternative to storing the form in a document, you can use the LotusScript Send method to design a form that you can mail along with a document. This ensures that the database has the correct form to display the document but doesn’t need to store the form with each document.

5.2.3 Views, folders, and navigators

The following design elements help organize the navigation inside the Domino application. Each of these elements are explained here:

- **View**: A view lists the documents stored in a Domino database and can be thought of as a “table of contents” of a database. Each row listed in a view represents data taken from a single document. Each column represents a field or a combination of fields taken from that document.

  All Domino databases have at least one view, but most of them have more. A view can display all the documents in the database, or it can display a subset of the documents. Documents can be viewed by categories, such as creation date or author. Views can present documents sorted on different fields, for example, sorted by topic.

- **Folder**: Folders enable you to store and manage related documents without putting them into a category, which requires a Categories field in the form used to create the documents. Folders are also convenient because you can drag documents to them.

- **Navigator**: A database navigator is a graphical interface that allows the user to easily access views, Domino data, or other applications. Navigators can include graphic buttons or hotspots, which are programmed areas that a user clicks to execute an action.

5.2.4 New R5.0 design elements

Domino R5 provides some new design elements. These elements are useful for building a more fashionable Web site. The new elements are:

- Pages
- Framesets
- Outlines
- Resources

For more detailed information, please refer to 6.1.1, “Overview of the new features in Domino Designer R5.0” on page 133.

5.2.5 Agents

Agents are standalone programs that perform a specific task in one or more databases. They can be run manually by the user, automatically when certain events occur such as mail arriving or documents being changed or added to the database, or scheduled to run at certain intervals. They can contain Notes simple actions, @function formulas, or a LotusScript or Java program. Agents are the most flexible type of automation because:

- They can be run by users in the foreground or run automatically in the background as scheduled agents.
- They aren’t associated with a specific design element.
- They can be run on a specific server, on several servers, on workstations, or the Web.
- They can call other agents.
- They can consist of simple actions, formulas, LotusScript, or Java programs.
- They can be distributed easily because they can be replicated.

- They can be personal or shared:
  - A personal agent is created and run by the same user. No one else can run a personal agent.
  - A shared agent is created by one user and can be run by other users.

Because agents are so flexible and powerful, you may want to consider their characteristics first to decide the type of agent you want to build and then build it.

The Agent Manager supports all aspects of building, running, and troubleshooting agents. The Agent Manager checks security, manages agent scheduling, monitors events and starts the appropriate agents when their associated events occur, records information in a log (the Agent Log), and performs database operations to run the automated tasks associated with the agent. Although you don't work directly with the Agent Manager, you use its components for building and troubleshooting an agent.

5.2.5.1 Where to use agents

Use agents for database-wide and domain-wide automated tasks and for complicated automated tasks. You can use them to easily access, process, and manage data on other servers, or in other databases.

> **Note**
>
> Some databases and templates have built-in agents. For example, the mail template has several built-in agents that let users manage their messages and customize their mail databases.

To use agents from the Web, you can use the following options:

- Use URL commands for opening agents. The syntax is:

  `http://Host/Database/AgentName?OpenAgent`

- Invoke them automatically by a WebQueryOpen or WebQuerySave event.
- Trigger them by server events, such as the arrival of new mail or on a scheduled basis.

To see the agents in a Domino database, select the database, and choose **View->Agents**.

5.2.5.2 Access control for agents

In the Access Control List (ACL) for the database, there is an option to Create Personal Agents. Since personal agents on server databases take up server disk space and processing time, the database manager may deselect this option to prevent users from creating personal agents.

The following options in the database access control list affect agents:

- To create a shared agent, a user must have Designer access or higher.
- To create a personal agent that is stored in a shared database, a user must be assigned the Create Personal Agents authority.
- To create a shared LotusScript or Java agent, a user must be assigned the Create LotusScript Agents option in the access control list. To store the agent

in a shared database, a user must also be assigned the Create Personal Agents authority.

When agents run, they automatically check the identity of a Domino user against any server document and ACL restrictions:

- When agents are run manually, they run with the identity of the Domino user.
- Scheduled agents run with the identity of the user who created or last modified the agent.
- A shared agent that runs from the Web uses the access rights of the user who created or last modified it. This default setting determines whether the agent can run and how much access it has to the server's file system.

  To add more security to a Web agent, force Domino to check the access rights of registered Web users:

  1. Choose **View**->**Agents**, and highlight the agent.
  2. Choose **Agent**->**Agent Properties.**
  3. Click the **Design** tab (the third tab from the left, as shown in Figure 44).
  4. Select **For Web Access: Run Agent as Web user**.

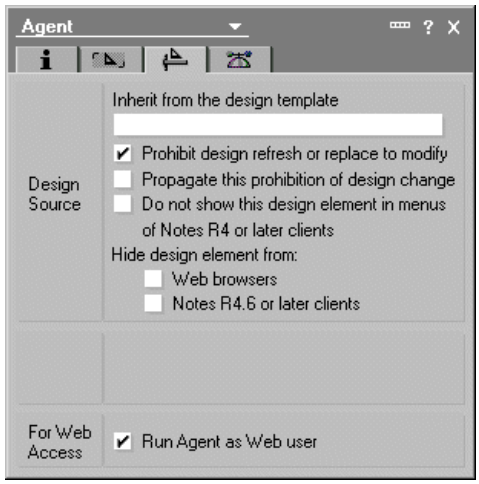


Figure 44. Agent's design properties

When Run Agent as Web user is selected, Domino prompts a Web user for their name and password when they attempt to run the agent (unless the Web user is already authenticated). Domino uses the login information to check for the invoker's rights in the database ACL.

Note

A Domino administrator can also specify restrictions in the server document to prevent users from running agents on a server. Users denied this server access cannot create agents on the server, regardless of the ACL setting.

5.2.5.3 Restricted and unrestricted agents

In the server document of the Domino Directory, you can determine who can run unrestricted or restricted LotusScript or Java agents. Using unrestricted agents, users have full access to the server's file system.

If the agent is written using LotusScript or Java, the agent signer must have the privilege of running this type of agent on the server for the Web user to run it.

This means Web users can run any agent, as long as the agent is not hidden from Web users. It is assumed the agent has been checked as being safe in a Web environment, by the last user who designed or modified it.

5.3 Languages

This section covers the available methods to use when programming in Domino. It discusses the entire range of programming languages available for use within Domino.

5.3.1 Formula language

Notes formulas are expressions that have program-like attributes. For example, you can assign values to variables and use a limited control logic. The formula language interface to Notes is provided through calls to @Functions. If you are familiar with the macro language in other products, such as Lotus 1-2-3, you will quickly become proficient in the @Functions in Lotus Notes.

@Functions are a powerful tool when you want to manipulate the current Notes document in an application. For example, you can use the following command to set the field Fieldname1 to the first name of the list NameList:

```
@SetField(Fieldname1; @Subset(NameList; 1));
```

Or, you can use only one function at a time, such as:

```
@UserName;
```

Error handling using formulas

New to Domino R5.0 are improved formula error messages. Formula error messages now give specific information as to where the error has occurred. This helps developers when debugging their applications. For example, if you are entering a syntactically incorrect @formula expression for field validation, Domino now gives you an accurate message as to where it has detected the error.

5.3.2 LotusScript

LotusScript offers you a wide variety of features of a modern, fully object-oriented programming language. Its interface to Notes is through predefined object classes. Notes oversees the compilation and loading of user scripts and automatically includes the Notes class definitions. This allows you to code your programs in an efficient way. While @Functions are ideal for coding simple logic, for example input translation or input validation of a field, LotusScript provides the ability to code loops, select (case) constructs, and do a lot more. Also, automatic indentation, which follows the program logic in IF-THEN-ELSE, and loop constructs is performed by the Integrated Development Environment (IDE) and makes your programs more readable and easy to maintain.

Furthermore, the hierarchy of the Notes Classes represents the flow of control you follow in the user interface if you step down from a database icon to a view, and further on to a document and to a specific field within this document. For example, if you are coding in LotusScript, you start with the UIWorkspace class

and go down to the UIDocument class, which represents the currently open document. Once you set this object variable, you have access to the fields of the document. The same principle applies if you are working with the backend classes of Lotus Notes, which represent the objects you may want to work with that are not in the user interface. You start at the NotesSession class and go down through the NotesDatabase class to the NotesDocument class.

Here is a short summary of the benefits offered by LotusScript:

- **Superset of BASIC**: Since LotusScript is a superset of the BASIC language, it is easy to learn, especially for Visual Basic users. You can write sophisticated scripts by using conditions, branches, subroutines, while loops, and others.
- **Cross-platform**: LotusScript is a multi-platform BASIC-like scripting language. Major platforms are supported, such as Windows, Macintosh, OS/2, and UNIX. You can create just one application, which can be used on any platform.
- **Object-oriented**: Domino Object Classes are available to LotusScript. You can write scripts to access and manipulate these objects. The scripts are event-driven, such as by an action, clicking the object or button, opening a document, or opening a view.
- **Included in Lotus applications**: Since LotusScript is supported by all the Lotus products, these products can access Notes classes using a Notes-supplied LotusScript extension. Another advantage is that you only need to learn one language to become proficient in writing scripts in other Lotus products.
- **OLE support**: Notes can be the perfect container for SmartSuite documents and other OLE-enabled applications, such as Microsoft Office. You can use external OLE 2.0 automation objects by scripting them, such as 1-2-3 worksheet objects. Notes registers itself as an OLE automation server. External applications can use these objects in scripts to create and reference them. LotusScript can combine all the parts and provide the means for controlling and manipulating objects.
- **Coexistence with Notes @Functions**: Lotus continues to support @Functions. LotusScript can work with them.
- **Integrated Development Environment**: The LotusScript Integrated Development Environment (IDE) provides an easy-to-use interface to create, edit, and debug scripts, and to browse variables and properties of classes. This allows you to write more complex scripts in Notes.
- **Extendable through LSXs**: You may extend LotusScript by writing your own classes, which are called LotusScript Extensions (LSXs). Creating your own LSXs allows you to expose custom functionality to LotusScript developers in precisely the same way as Notes functionality is exposed. You might use this, for example, if you have customer processing logic, such as a proprietary pricing process, that you wanted to make available to Notes developers. Actually, there are a lot of LSX packages from Lotus company, such as LSX:DO, LSX LC, Lotus LSX for R/3, MQ LSX, and so on. You can use these package to access your enterprise data in the relational database.

Note

Functions or subroutines written in LotusScript cannot be called directly from the Web browser. If you want to use LotusScript in a Web environment, you should use agents. You can refer to 5.2.5, “Agents” on page 102, for more details about agents.

5.3.3 Java

As the Web evolves, Java becomes a more important and more commonly used programming language. Domino offers you the option to write your applications in Java. For example, you can write your Domino agents in Java. Domino supports Java programs written in Java 1.1.x. Where you use Domino R5.0 on the AS/400 system, you can use the JDK 1.1.6 or 1.1.7, provided on the AS/400 system in the AS/400 Developer Kit for Java (5769JV1).

Note

If running OS/400 V4R4, you must install the 5769JV1 base and at least one of the optional components:

- Option 1: Java Developer Kit 1.1.6
- Option 2: Java Developer Kit 1.1.7

For servlets running environment, Domino R5.0 use JSDK 2.0.

5.3.3.1 About Java Domino classes

Java Notes classes are created by modifying some of the LotusScript Extension (LSX) architecture to include a Java “adapter” to compose the new Java Domino classes. The Java Domino classes have similar functions to some of the LotusScript Domino backend objects. You can use these classes from any Java program (within the Notes Designer environment or outside of it) as long as Notes Release 5 is installed on the machine.

Internally, Java Notes classes execute the same C++ code as the LotusScript Domino backend objects. Only the language syntax is different. A Java program is generally made up of a number of files. You must designate one of the files as the Base Class, which is the starting point for the Java program. For efficiency, typically for improving applet download speeds, you can bundle all of the class files and additional resources (for example GIF files) into a single compressed Java Archive file. The imported Java files can be of the following types:

- Class: *.class
- Archive: *.jar

For example, when you write a Java agent program, the class you write must extend the class *lotus.notes.noiAgentBas*e. The code you want to execute when the agent runs is in the *NotesMain()* method.

5.3.3.2 Java coding conventions

There are some conventions you should follow to write a Java program.

Classes

The names of the Java classes are the same as for LotusScript, except they begin with the “Notes”-prefix. Table 13 shows how some of the Java Domino classes correspond to LotusScript objects.

Table 13. Java Domino class

Java class	LotusScript object
Lotus.Notes.Session	NotesSession
Lotus.Notes.DbDirectory	NotesDbDirectory
Lotus.Notes.Database	NotesDatabase
Lotus.Notes.View	NotesView
Lotus.Notes.Document	NotesDocument
Lotus.Notes.Item	NotesItem
Lotus.Notes.RichTextItem	NotesRichTextItem

Note

By convention, you should start your own classes with the first character in uppercase. Because Java is case sensitive, using the wrong case will cause an error.

Methods

Method names are written with the first character in lowercase, for example getFirstDocument. There are some exceptions, such as FTSearch.

Properties

To access properties in Java, you also have to use methods. In Java, properties are implemented through methods, known as accessors, which use the following naming conventions:

- The name of a method used to get the value of a non-boolean property is the name of the property prefixed with “get”.
- The name of a method used to set the value of a property is the name of the property prefixed with “set”.
- The name of a method used to get the value of a boolean property is the name of the property prefixed with “is”.

Parameters and return values

Parameter and return values differ from LotusScript as needed to match the different data types in Java. For example, in LotusScript boolean values are represented by Variants. In Java, they are of type boolean.

Object containment hierarchy

In Java, you cannot create lotus.notes objects using the “new” modifier. All lotus.notes objects must be created with lotus.notes methods emanating from the root Session object.

5.3.3.3 Agents, applets, applications, and servlets

Java programs can take one of several forms, each with its own characteristics. The differences between these forms are summarized here:

- **Java agents**: Complement the familiar LotusScript agents. To a large degree, they can be used interchangeably when dealing with backend operations. Some reasons for choosing Java over LotusScript are:
  - Existing programmer knowledge
  - Multi-threading
  - A more fully featured language
  - Extensibility through (non-visual) beans
- **Applets**: Allow a Notes developer to create a richer GUI environment for the end user. Applets are dynamically downloaded from the server and executed on the client's machine and work with either Web browsers or Notes clients. The functions of applets can vary widely, from simple news tickers to complex database front ends. Java applets are subject to the Java Sandbox security model, which prevents unauthorized applets from accessing sensitive machine resources and from performing certain operations. You do not have to distinguish between local and remote access in the main code of an applet. AppletBase makes local calls if the applet is running through the Notes client and remote (IIOP) calls if it is running through a browser.
- **Java applications**: Differ from applets in that they are not dynamically loaded from the server. They are similar to traditional executables in this respect. However, Java applications typically run outside of the Java "Sandbox" security model and can, therefore, access machine and network resources denied to an applet. A Java application can make local and remote access to a Domino database.
- **Java servlets**: As their name suggests, servlets only run on the server. A servlet is invoked by a client request and will respond directly to the client. Typically a servlet is used to provide a high performance link to a backend system and format the results back to the client as a HTML document. However, servlets are not restricted to serving, just HTTP requests. In fact, servlets converse directly with any suitable client application (usually an applet). Servlets can make local and remote access to Domino database.

5.3.3.4 CORBA support

One of the major enhancements in the Domino R5.0 embrace of Internet standards is the support for Common Object Request Broker Architecture (CORBA). CORBA is an open standard defined by the Object Management Group (OMG). CORBA serves as middleware for a distributed computing environment, where clients can invoke methods on remote APIs residing on other computers. It uses Internet Inter-ORB Protocol (IIOP) for communication over a TCP/IP network. CORBA/IIOP enables information to be processed efficiently over networks within an open standards-based framework and to distribute work effectively between clients and servers, ultimately lowering the cost of ownership.

Domino R5 unveils support for CORBA, so you can build robust, distributed applications. The ORB technology and Java allow you to create client applications that are dynamically loaded from the server with transparent access to the server-side Domino objects, for example, to initiate a workflow process. In addition, while Notes client applications have been able to access Domino objects for quite some time, CORBA and IIOP support in Domino R5 expand this

access to Web clients. Your primary access to this ORB is through Java applets or applications. For example, you can place a custom Java applet on a form and have that applet access objects in either the Notes client or a Web browser. For the Notes client, you are actually using the local Java interfaces. For the browser, you are using the CORBA-remote objects. That is, the applet uses IIOP to connect back to the ORB on the server.

CORBA allows you to create client-side objects that “talk” IIOP across the wire to the server-side ORB, which is hard-wired to Domino’s backend classes for better performance. The main purpose is to off load the server by projecting its services to the client. Browser applications can then run locally with the context of the Domino server. For example, you can interact with your server-based mail locally, without involving the server in each user transaction.

With support for CORBA and IIOP, Domino allows you to create client/server Web applications that take advantage of the Domino objects and application services. In addition, you can now access backend relational databases for enhanced data integration using DECS. In previous releases, when you designed a Web application, the Domino objects classes (formerly known as remote backend classes, or the Notes Object Interfaces or NOI) allowed you to access data that was not on display in the browser. These backend classes were LotusScript or Java objects. In R5, these objects are available to the browser using CORBA, as shown in Figure 45.

This means that now your Web application is similar to your Notes application in terms of programmability. In a Domino application, you could always manipulate data that was on display and data in other databases. In a Web application, you had to wait for a user to open or save a Web page to access this same data. CORBA allows you to access the data without the user opening or saving the document from their workstation.

You can also embed a CORBA applet in a document or a form using the same procedure as for any other applet. You can use a browser to view embedded CORBA applets on a Domino server. It is no longer necessary to set alternate HTML. A CORBA property box setting tells Domino to provide the HTML source that the applet needs to make an IIOP connection back to the server.

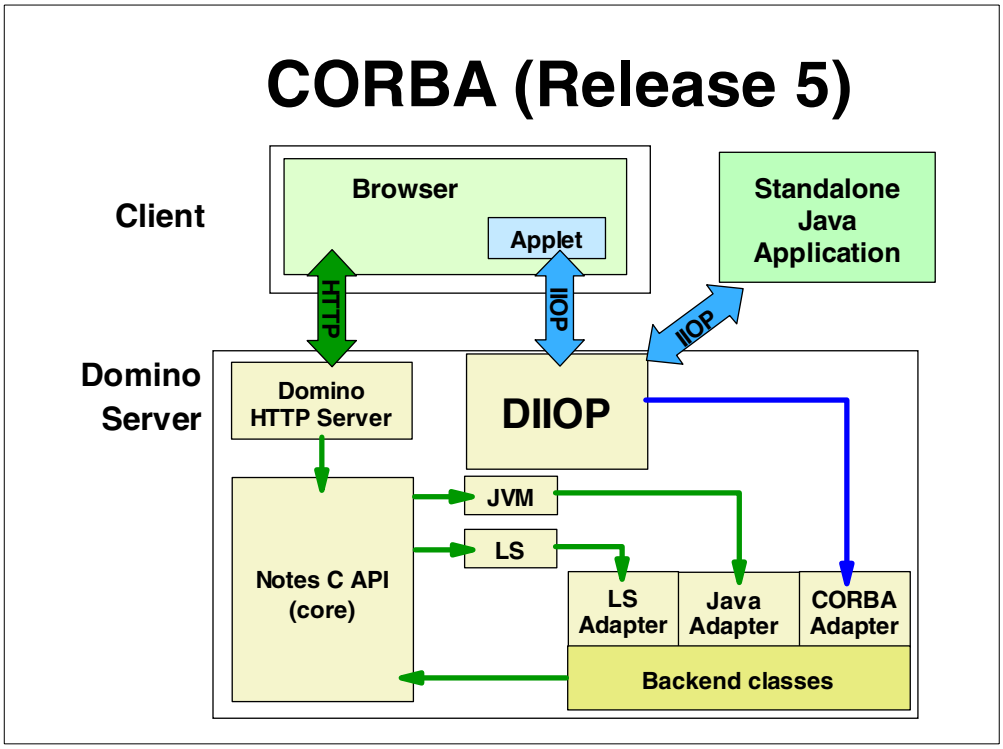


Figure 45. Domino CORBA architecture

5.3.3.5 CORBA implementation

Domino R5 uses CORBA to access Java programs on remote clients such as applets in browsers and standalone Java applications to access the Domino objects on the Domino server. From an implementation standpoint, a remote client instantiates and references Domino objects as if they reside at the client. In fact, these objects are instantiated at the Domino server. When the client references these objects, it is actually communicating with the objects on the server. This is seamless to the programmer (Figure 46 on page 112).

Java Client IDL

There is one Interface Description Language (IDL) definition file per Domino object C++ class that is exposed to CORBA. IDL files are published for developers to create their own stubs. The Java Client IDL stubs are contained in the lotus.domino package in the NCSO.jar file. This jar file also contains the Java Client ORB classes.

The jar file is automatically loaded down to a browser client if it is designated as a CORBA applet through the properties box.

Domino R5 Java Client ORB

The Domino R5 Java Client ORB is essentially a complete Server ORB but has been stripped down and compressed. It also has enhanced security, allowing clients to use SSL to create authenticated sessions with the server. The Java Client ORB classes are contained in the NCSO.jar file. Currently, the ORB is instantiated once per getSession() method invocation. Two applets can have more than one instantiation of the ORB per HTML page. Such technologies as InfoBus can fix this situation by sharing the same Session object reference.

Domino R5 Server ORB implementation

The Server ORB is the “broker” that receives requests from objects to access other objects. It functions as a sophisticated router that passes the requested information to the requested object. It also passes information back to the requester, as necessary.

The R5 ORB allows Domino objects to load and respond to client IIOP requests. Its primary use is for doing client-side processing in Domino Web applications. The R5 ORB is a modified version of the IBM ORB. A significant portion of the original IBM ORB has been stripped out, including the Interface Repository. Improvements have also been made to address such issues as scalability. Other customer created ORBs may also run on the Domino server.

The two elements that Lotus eliminated from the ORB (by hard-wiring) are the Location Service and the Name Service. Other than that, it is a standard CORBA object server. This Server ORB process can be loaded at server startup by placing it in the NOTES.INI file:

```
ServerTasks =<other tasks>,http,diiop
```

The Server ORB listens to IIOP requests through a different port than HTTP. This port can be changed through the Server Document.

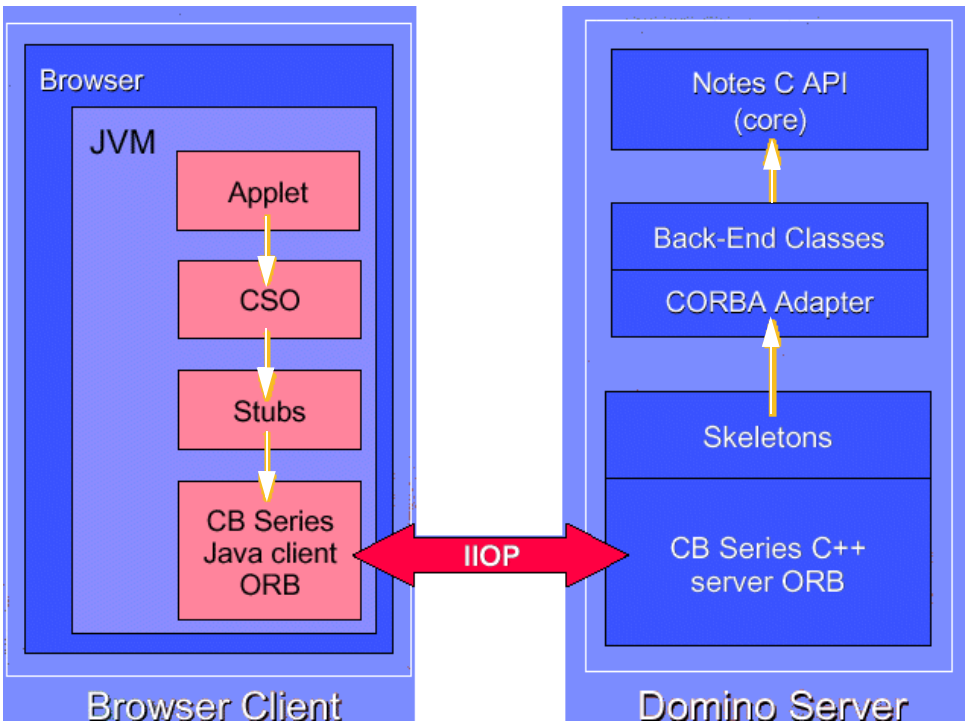


Figure 46. Domino CORBA implementation

5.3.3.6 Domino Java classes

You can access Domino objects using Java programs since Domino Release 4.6. However, support has changed with Domino R5. In the following sections, we explain what can be done with both releases.

Domino R4.6
In R4.6, the Domino Java library is provided with the Notes.jar file, which contains the lotus.notes classes—the notes object interface (NOI) in R4.6 terminology. Using the NOI classes, you can create Java programs that access named databases, views, documents, and other backend Domino objects. However, you need to run these Java programs on a machine where the NOI classes and a Domino client are installed.

The Java library allows the same functionality as the LotusScript interface. You can benefit from additional advantages provided by Java, including:

- Multithreading
- Standalone application development
- Easy network access

The Domino R4.6 NOI classes
The Domino R4.6 NOI classes:

- Include only backend classes
- Include properties that are accessed through methods

Domino R5
In Domino R5, the NOI have been renamed to Domino Objects. DominoR5 uses CORBA to allow remote Java programs to access remote Domino objects on a Domino server. The client Java program (a standalone application or an applet), through IIOP, talks over the network to a Domino server automatically using the same DOM objects.

In R5, the Domino Java library is provided with the following packages:

- **Notes.jar**: Notes.jar contains the high-level lotus.domino package, the lotus.domino.local package for local calls, and the R4.6 lotus.notes package to provide backward compatibility with applications written with Domino R4.6.
- **NCSO.jar**: NCSO.jar contains the lotus.domino package for CORBA. The package, used by remote Java applications and applets only, contains the high-level lotus.domino package and the lotus.domino.corba package for remote calls.

With reference to the CORBA architecture, the Domino Java library contains the Java client IDL stubs and the Java client ORB classes.

The server ORB is the broker that receives requests from objects to access other objects. It functions as a sophisticated router that passes the requested information to the requested object. It also passes information back to the requester, as necessary.

The Domino R5 ORB allows DOM objects to load and respond to client IIOP requests. Its primary use is for doing client-side processing in Domino Web applications.

5.3.4 JavaScript

Before Domino R5.0, the main use of JavaScript within Domino was to modify the standard behavior of Web pages by adding some “client side” functions. Client side means that no “server side” activity is requested by JavaScript functions, so that their result is presented quickly to the end user. With Domino R5.0, you can

use JavaScript to write applications which will support both the Notes client and the Web browser. All events associated with an object are programmable, using JavaScript, LotusScript, or even simple @functions, and can be easily accessed within the Programmer's Pane.

JavaScript allows you to handle such events as onLoad (for a Web page), onClick (for an input button on form), onChange, onBlur, onFocus (for input fields), and so on. You can use these events to trigger JavaScript functions that can also perform some complex operations. In a Web browser, JavaScript functions can access all elements on a Web page (like input fields), as well as properties and methods that control the status and the behavior of the Web browser window itself. Adding JavaScript to Domino forms and fields is particularly useful, as it allows you to create forms with a more dynamic behavior without adding workload to the Domino server. For example, with JavaScript, field values can be validated locally on the browser, instead of on the Domino server, after submitting.

5.3.4.1 Using JavaScript in Domino design elements

To use JavaScript in your application, you add JavaScript code to events as you do with LotusScript. Table 14 shows some of the supported JavaScript events for forms and pages.

Table 14. JavaScript events

Notes form event handlers	Description
onLoad	The cursor moves over the form or page.
onUnLoad	Before the document is cleared.
onSubmit	Window event, before the document is saved.
onReset	Window event, before the document reset.
onBlur	When the forms loses focus.
onClick	An object on a form is clicked.
onDblClick	The user double-clicks a form element or a link.
onFocus	The form receives focus.
onHelp	Triggered when the user presses the F1 (Help) key.
onKeyDown	The user presses a key down.
onKeyPress	The user presses or holds down a key.
onKeyUp	The user releases a key.
onMouseDown	The user presses a mouse button down.
onMouseMove	The user moves the cursor.
onMouseOut	The cursor leaves the form or page.
onMouseOver	The cursor moves over the form or page.
onMouseUp	The user releases a mouse button.

If you want to add JavaScript code for other Window events that are not handled by the Notes client (such as onFocus, onResize, or onMove), you can do this in the HTML Body Attributes object for the form. Likewise, if you want to add an event for a form element, you can do it in the HTML Body Attributes Field event. Code in pass-thru HTML and the HTML body attributes fields is passed to the browser, but ignored in the Notes client. JavaScript must be enabled in the User Preferences to be executed by the client.

Note

If you enter JavaScript code into a formula, keep the following rules in mind:

- Within the text string that you are going to put in the formula, every double quote ("), single quote (’), and backslash (\) must be preceded by a backslash (\).

  For example, the following text string:

  ```
  <a href="http://www.ibm.com"> click here </a>
  ```

  must become:

  ```
  <a href=\"http://www.ibm.com\"> click here </a>
  ```

- The same text string must be included between two double quotes before pasting it into the formula, for example:

  ```
  "<a href=\"http://www.ibm.com\"> click here </a>"
  ```

5.3.4.2 The browser JavaScript object hierarchy

The JavaScript object hierarchy exposed in a browser appears as shown in Figure 47.

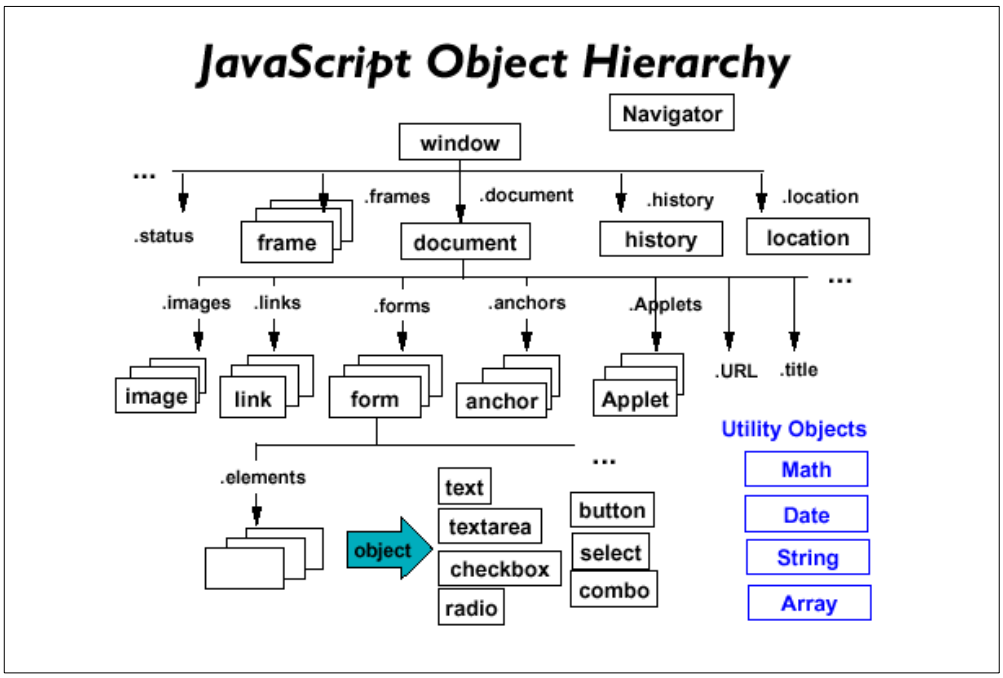


Figure 47. JavaScript object hierarchy

Some of the object relationships illustrated above do not make much sense in the Notes client, for example, not having more than one form attached to a document at any one time. Therefore, the Domino JavaScript object hierarchy looks like the example in Figure 48.

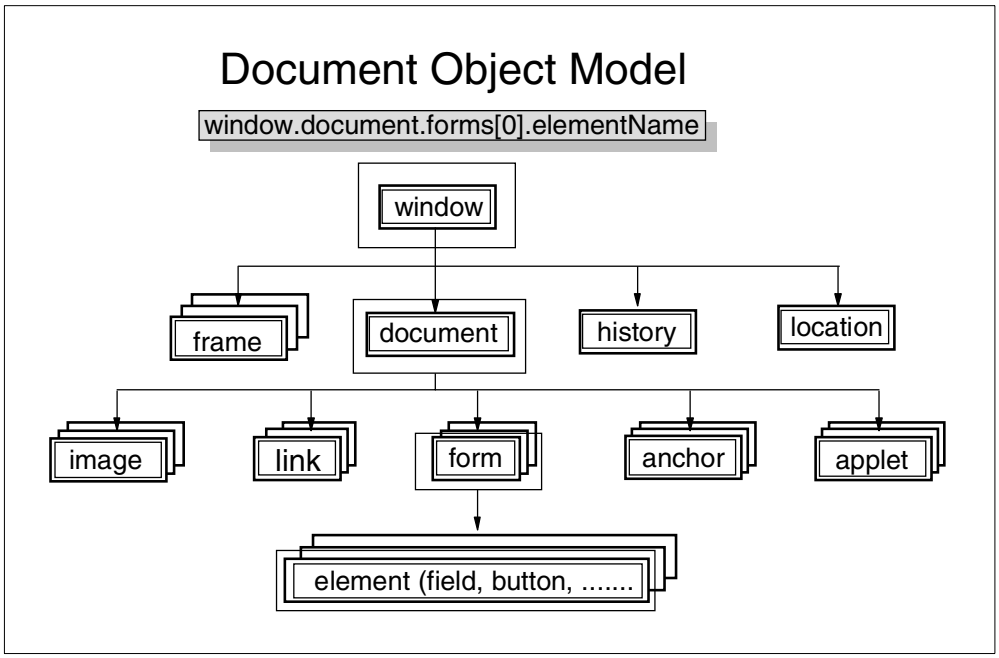


Figure 48. Notes JavaScript object hierarchy

5.3.5 XML

Extensible Markup Language (XML) is a standard for creating markup languages that describe the structure and meaning of data in a document. XML separates the content of a document from its presentation and provides a common format for transferring data across the World Wide Web (WWW) or company intranet. The result is a technology that makes data available regardless of the proprietary systems involved.

XML is not a markup language itself, but a set of rules that enables you to create the tags you need as you need them. In XML, a document is broken into parts, and each part is separate. The way a document looks is determined by a style sheet; the content of a document is contained in a separate file; and the definitions of the tags are stored in another file called a Document Type Definition (DTD). Using a common data format and separating data from its formatting and its tag definitions makes the data readily accessible and reusable. Data from one application can be used in another application by changing the DTD or style sheet.

There are several ways you can include XML in a Designer application and serve the data to an XML parser:

- You can enter XML tags that describe data on a form or a page. By treating the contents of the form or page as HTML, you can serve the XML to an XML parser that can interpret the tags. XML describes the data being presented. To format and style the data on the form or page, you can use a stylesheet, created with the Extensible Stylesheet Language (XSL) to transform the data

into HTML, or you can use a Cascading Style Sheet (CSS) to style the XML directly on the client.

- You can also generate XML data with a view by including XML tags in column formulas. To pass the view to the server, you must embed it on a page or view to wrap the whole view in the correct XML document definition tags.
- You can use agents or servlets to dynamically generate or store XML. Agents are useful for running a scheduled process in a Domino application. Servlets run on the server based on a request from a Web browser.

5.3.5.1 Current support for XML in Domino

Today, developers have several ways to produce XML from a Domino database or to pull XML into a Domino database. They can:

- Use Domino views to export XML using the `?ReadViewEntries` URL syntax.
- Write LotusScript or Java agents to produce XML or to import it.
- Use Java servlets to produce or import XML or to transform XML to another format, using the Lotus XSL Processor.

5.3.5.2 Future support for XML in Domino

In Release 5.0.3, Lotus will add one more URL syntax, ?ReadNote, which will open a note and return its contents in XML format.

In Release 5.0.3, you will also see the first appearance of XML parsing and transformation engines in the product. Lotus will integrate these into the Java backend classes so that developers can make use of these foundation services in a way that is really natural for Domino applications.

5.4 Tools for enterprise integration

Most organizations have many computing systems, with Lotus Notes/Domino being only one piece. Since it is important to easily access data from all these systems and to build business applications based on that data, Lotus has endowed Domino R5 with excellent tools for doing just that. External data now can easily be accessed and integrated into Domino applications without the need for add-on products or a lot of tricky programming. This section describes the options available for Domino for AS/400 integration with enterprise applications.

5.4.1 @DB functions

One way to program logic in a Domino application is using formulas. Notes Formula Language is a macro-like language. A function is called using a single statement (line of code). @DB functions provide the ability for both Notes clients and Domino servers to use data stored in Domino databases and relational databases. The @DB formulas are read-only.

@DBColumn, @DBLookup, and @DBCommand are Notes functions that enable you to access RDBMSs which use the underlying ODBC interface. @DBColumn and @DBLookup are available for a Domino application to access other Domino databases (or even data in the same Domino database). @DbColumn and @DbLookup only retrieve data from one table at a time.

@DbCommand retrieves data but does not work with result sets. It only retrieves the first column in a SELECT SQL command that specifies multiple columns.

The basic purpose of these functions is to create value lists for keyword fields. @DBColumn and @DBLookup can be used to query a relational database. @DBCommand is used only for executing stored procedures, but does not return result sets. If you need a more customized and more complex query, LS:DO is a better option.

The @DB functions are summarized in Table 15.

Table 15. Summary of @DB functions

Functions	Descriptions	Equivalent SQL
@DBColumn	Generates a keyword list. Returns a specified column for all rows in the specified table.	SELECT DISTINCT column_name FROM table_name
@DBLookup	Performs a lookup. Returns a specified column value in the row that matches the specified condition.	SELECT column FROM table WHERE condition
@DBCommand	Triggers stored procedures in the external database	(any SQL statement)

5.4.1.1 @DBColumn

The following syntax is for @DBColumn:

```
@DBColumn("ODBC": Cache ; DataSource ; UserID1 : UserID2 ; Password1 :
Password2 ; TableName ; ColumnName : NullHandling ; Distinct : Sort )
```

The following example formula gets the PARTNO column of the MANUALS table:

```
@DbColumn("ODBC";"TESTDOMINO";"ODBC";"TESTDOMINO";"";"ODBC";"TESTDOMINO";"ODBC
";"TESTDOMINO";"";"";"MANUALS";"PARTNO";"MANUALS";"PARTNO";"":"Ascending")
```

The @DbColumn parameters are described in further detail here:

"ODBC" Keyword. Indicates that you are accessing an ODBC data source.

"NoCache" Keyword. Optional. If you want to ensure that Notes/Domino retrieves the latest information for every lookup, specify this option, as in “ODBC”:”NoCache.” Omit NoCache if you want the results of the lookup to be *cached*, that is, stored for re-use. Each subsequent lookup to the same location within the same Notes/Domino session re-uses that information, as long as the database executing this lookup remains open.

If you omit “NoCache”, you do not need to replace it with anything. The lookup results are cached automatically, but you may specify “Cache” for readability.

data_source Text. The name of the external data source being accessed. A data source indicates the location of one or more database tables.

user_ID1: user_ID2

Text list. The user IDs needed to connect to the external database. You may need up to two IDs, depending on the DBMS being accessed.

password1 : password2
Text list. The passwords required by the user IDs.

table Text. The name of the database table being accessed.

column Text. The name of the column from which data is being retrieved.

null_handling Text. Specifies how null values are treated when the data is retrieved.

"Distinct" Keyword. Optional. Removes duplicate values from the list before returning data.

sort Keyword. Specify “Ascending” to sort the list of values into ascending order before it is returned; specify “Descending” to sort the list in descending order.

5.4.1.2 @DBLookup

The following syntax is for @DBLookup:

```
@DBLookup( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ; Password1 :
Password2 ; TableName ; ColumnName : NullHandling ; KeyColumn ; Key ; Distinct
: Sort )
```

The following formula gets the TITLE from the row of the MANUALS table where PARTNO is 17-895A:

```
@DbLookup("ODBC";"TESTDOMINO";"ODBC";"TESTDOMINO";"";"";"MANUALS";"TITLE";"PAR
TNO";"17-895A")
```

The @DbLookup parameters are described in further detail here:

"ODBC" Keyword. Indicates that you are accessing an ODBC data source.

"NoCache" Keyword. Optional. If you want to ensure that Notes/Domino retrieves the latest information for every lookup, specify this option, as in “ODBC”:”NoCache.” Omit “NoCache” if you want the results of the lookup to be cached, that is, stored for re-use. Each subsequent lookup to the same location within the same Notes/Domino session re-uses that information, so long as the database executing this lookup remains open.

If you omit “NoCache”, you do not have to replace it with anything. The lookup results are cached automatically, but you can specify “Cache” for readability.

"data_source" Text. The name of the external data source being accessed. A data source indicates the location of one or more database tables.

"user_ID1" : "user_ID2"
Text-list. The user IDs needed to connect to the external database. You may need up to two IDs, depending on the DBMS being accessed.

"password1" : "password2"
Text list. The passwords required by the user IDs.

"table" Text. The name of the database table being accessed.

"column" Text. The name of the column from which data is being retrieved.

"null_handling" Text. Specifies how null values are treated when the data is retrieved.

"key_column" Text. The name of the column used for key matching.

"key" Text, number, or date-time, or a list. The value to be looked up in key_column. Use the Notes/Domino type that agrees with the type of the key column in the data source.

"Distinct" Keyword. Optional. Removes duplicate values from the list before returning data.

"sort" Keyword. Sorts the list of values into either ascending or decending order before it is returned.

5.4.1.3 @DBCommand

The following syntax is for @DBCommand:

```
@DBCommand( "ODBC": Cache ; DataSource ; UserID1 : UserID2 ; Password1 :
Password2 ; SQL ; NullHandling )
```

The following formula gets the PARTNO column value for every row of the MANUALS table where the numeric value in the ONHAND column is less than 100:

```
@DbCommand("ODBC";"TESTDOMINO";"ODBC";"TESTDOMINO";"";"";"SELECT PARTNO FROM
MANUALS WHERE ONHAND <100")
```

The @DBCommand parameters are described in further detail here:

"ODBC" Keyword. Indicates that you are accessing an ODBC data source.

"NoCache" Keyword. Optional. If you want to ensure that Notes/Domino retrieves the latest information for every lookup, specify this option, as in "ODBC":"NoCache." Omit "NoCache" if you want the results of the lookup to be cached, that is, stored for re-use. Each subsequent lookup to the same location within the same Notes/Domino session re-uses that information, as long as the database executing the lookup remains open.

If you omit "NoCache", you do not have to replace it with anything. The lookup results are cached automatically, but you can specify "Cache" for readability.

DataSource Text. The name of the external data source being accessed. A data source indicates the location of one or more database tables.

user_ID1 : user_ID2
Text list. The user IDs needed to connect to the external database. You may need up to two IDs, depending on the DBMS being accessed.

password1 : password2
Text list. The passwords required by the user IDs.

command_string Text. An SQL statement, command statement, or name of a procedure to be executed.

null_handling Text. Specifies how null values are treated when the data is retrieved.

5.4.2 LotusScript: Data Object

The LotusScript: Data Object (LS:DO) provides full read and write access to external ODBC data sources using the control and flexibility of a structured programming language: LotusScript. LS:DO is excellent for real-time data access from any LotusScript event in Notes, such as clicking a button, exiting a field, or opening a document.

LS:DO is available on both the Notes client and the Domino server. In most cases, an ODBC driver is needed on either the server or the client workstation. For Domino for AS/400, if these functions run on the server, they have a direct access to DB2 UDB for AS/400 using the DB2 Call Level Interface (CLI).

Figure 49 shows a schematic representation of the components in the LS:DO framework that allow a Notes application to access a database.

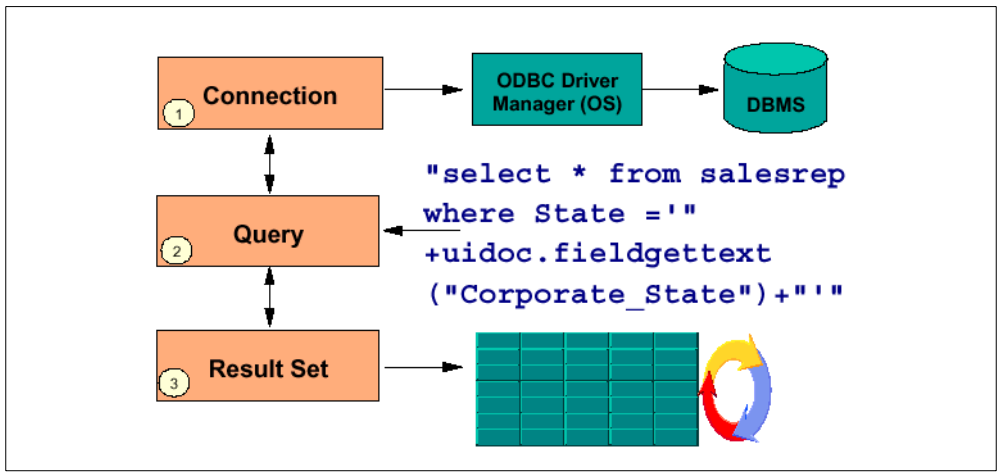


Figure 49. General LS:DO framework

LS:DO consists of a set of three classes: ODBCConnection, ODBCQuery, and ODBCResultSet. These classes come complete with a powerful set of properties and methods and full SQL capabilities. LS:DO is a high-level abstraction of the ODBC feature.

5.4.2.1 LS:DO classes

The LS:DO classes are:

- **ODBCConnection**: The ODBCConnection class allows you to establish a connection and to access database catalog information, such as data source lists, table lists, procedures lists, and so on.
- **ODBCQuery**: The ODBCQuery class is used to hold the ODBCConnection object in which a connection is established, and to hold the SQL statement you use to perform the inquiry. The SQL statement is parsed through the ODBC driver that your application requires.

- **ODBCResultSet**: The ODBCResultSet class has the functions used to handle records that are termed result sets. A result set holds the retrieved records of an SQL query, which is specified with the ODBCQuery object.

5.4.2.2 Where to use LS:DO

LS:DO is best suited to handle the following situations:

- **LotusScript programming environment**: If you develop an application with the LotusScript environment, you can easily use ODBC access through LS:DO classes.
- **Low-volume data transfer**: LS:DO is more suited for low-volume access to data resources. From a performance perspective, LS:DO is not well suited to moving large volumes of data.
- **Easy data access**: When your application needs to read and update data in an RDBMS, LS:DO provides an easier way than the ODBC API or the @DBCommand because of the classes that allow you to work with result sets.
- **Real-time direct access**: LS:DO is integrated directly in a Notes application and so on.

5.4.2.3 Additional considerations

The following statement must be specified in the Define (Globals) Event (Declarations) within Lotus Notes:

```
Uselsx "*LSXODBC"
```

On the AS/400 system, you must specify the user ID and password in your script. You must also specify the collection name (or library name). The name of the table must be: collection_name.table_name.

Using and asterisk (*) in the SQL select statement should be avoided. File names should be specified.

Result.Close(DB_CLOSE) must be used to close the connection to the data source

5.4.3 LotusScript extension for Lotus Domino Connectors

A LotusScript Extension called LotusScript Extension for Lotus Domino Connectors (LSX LC) has been available since Domino Release 4.6.3. LSX LC allows programmatic access to data external to Domino applications.

LotusScript provides an integral programming interface to Lotus Notes. The LotusScript Extension for Lotus Domino Connectors enhances the power of Notes by extending its scripting capabilities to data outside of Notes.

LSX LC is based on the Lotus Domino Connectors. It makes the coding more independent of the type of data source.

5.4.3.1 LSX LC classes

The Lotus Domino Connectors provides external data and system access to the Domino LotusScript environment. The LSX LC classes consist of *LCConnection*, *LCFieldlist*, and *LCField*, and four advanced data types: *LCStream*, *LCNumeric*, *LCCurrency*, and *LCDatetime*. In addition to these seven classes, there is also an

LCSession class. Each of these classes and their primary usage is described in the following list:

- **LCSession Class**: The LCSession class provides error information useful in error handlers. It also provides for the query and lookup of available Lotus Connectors.
- **LCConnection Class**: The LCConnection class represents an instance of a Lotus Connector. This class provides query and data access to the external system. Multiple connections can be allocated to a single Lotus Domino Connector.
- **LCFieldlist Class**: The LCFieldlist class is the primary class for manipulating data through a connection. It binds a group of fields together with names and an implied order. Fieldlists are used primarily for a number of connection operations, result sets and selection criteria, as well as reading and writing data.

  When a result set is generated, and an empty fieldlist is initially passed in, the field-test is automatically populated by the connector. For each data element of the result set, the field-test receives the element name and a field object of the corresponding data type. The result set can be controlled by manually building the fieldlist before the result set is constructed or by using the FieldNames property of the connection.

  The Select and Call connection methods use an optional fieldlist of keys or parameters to restrict the result set. This fieldlist is manually constructed and passed to the select method. A key or parameter list is constructed by appending or inserting names and data types to the list. These methods create fields in the fieldlist and return these fields for further manipulation. These fields are then given values and, using field flags, can be given conditions such as greater-than, not-equal, and so on.
- **LCField Class**: The LCField is the storage class that contains one or more data values. The data type of a field is for all values contained within and may be any of the four advanced data types discussed in the following topics, as well as a long integer and double-precision floating point, and, in some advanced usage, fieldlist, or connection.
- **LCStream Data Type**: LCStream is a general purpose text and binary data type. The contents of a stream are marked with a format that details the character set of the text or any special attributes of the binary data.
- **LCNumeric Class**: The LCNumeric class is a container for very high precision numbers.
- **LCCurrency Data Type**: The LCCurrency class is a fixed point decimal data type with four decimal places and 19 digits of precision. This is mathematically equivalent to the LotusScript data type and is provided to support connections with a dedicated currency.
- **LCDatetime Data Type**: The LCDatetime class is a date and time data type that is accurate to the hundredth of a second and is aware of time zones and daylight savings time.

5.4.3.2 Where to use LSX LC

In an AS/400 environment, LSX LC can be used in the following environments:

- In an application running on the Domino for AS/400 server (Web serving or agent)
- In an application (or function) running inside the Notes client (Release 4.6.3a or later)

Wherever the script using LSX LC is running (Domino server or Notes client), you must load the LotusScript Extensions for Lotus Connectors specifying the `UseLSX "*lsxlc"` statement in the script.

5.4.4 Java integration options

Several options to integrate DB2 UDB for AS/400 and AS/400 applications are available in a Domino for AS/400 environment. Some of them are presented here:

- Java Database Connectivity (JDBC)
- AS/400 Toolbox for Java
- CORBA

5.4.4.1 JDBC

The *JDBC API* defines Java classes to represent database connections, SQL statements, result sets, database metadata, and so forth. It allows a Java programmer to issue SQL statements and process the results. JDBC can handle dynamic SQL and stored procedures calls. JDBC is the primary API for database access in Java.

The JDBC API is implemented through a driver manager that can support multiple drivers connecting to different databases. JDBC drivers can either be written entirely in Java so that they can be downloaded as part of an applet, or they can be implemented using native methods to bridge to existing database access libraries.

JDBC provides access to DB2 UDB for AS/400 data wherever Java is used with a TCP/IP connection to the AS/400 system. JDBC can also be used in Java processes running on the AS/400 system. This includes Domino and Notes agents, and Java processes (servlets and applications) running directly on the AS/400 system.

Two JDBC drivers are available on the AS/400 platform:

- The DB2 JDBC driver (recommended on the AS/400 system)
- The AS/400 Toolbox for Java JDBC driver (server and client)

5.4.4.2 AS/400 Toolbox for Java

The *AS/400 Toolbox for Java* allows access to AS/400 objects wherever Java is used on a LAN-connected client or on the AS/400 system. This includes Java agents running on a Notes client or on the Domino server, as well as Java applications and servlets. The Toolbox consists of a set of Java classes that provide a programmatic interface to a variety of OS/400 object types, including programs, commands, user spaces, and data queues, as well as relational data in DB2 UDB for AS/400.

The following system resources are supported for DB2 UDB for AS/400 access:

- **JDBC database access**: DB2 UDB for AS/400 data can be accessed using a JDBC driver written to the interface defined by the JavaSoft JDBC specification.
- **Record-level database access**: AS/400 physical and logical files can be accessed a record at a time using the interface of these classes. Files and members can be created, read, deleted, and updated.

5.4.4.3 CORBA and Domino

With support for CORBA and IIOP (described in 5.3.3.4, “CORBA support” on page 109), Domino now allows you to create client/server Web applications that take advantage of the Domino objects and application services. In addition, you can now access backend relational databases for enhanced data integration using the new Domino Enterprise Connection Services. In previous releases, when you designed a Web application, the Domino Object Model (formerly known as the remote backend classes, or the Notes Object Interfaces or NOI) allowed you to access data that was not on display in the browser. These backend classes were LotusScript or Java objects. In R5.0, Lotus has made these objects available to the browser using a technology called Common Object Request Broker (CORBA).

This means that now your Web application is largely similar to your Notes application in terms of programmability. In a Notes application, you could always manipulate data that was on display and data in other databases. In a Web application, you had to wait for a user to open or save a Web page to access this same data. CORBA allows you to access the data without the user opening or saving the document. Instead, you can use Java or JavaScript on a W3C event.

You can also embed a CORBA applet in a document or a form using the same procedure as for any other applet. You can use a browser to view embedded CORBA applets on a Domino server. It is no longer necessary to set alternate HTML. A CORBA property box setting tells Domino to provide the HTML source that the applet needs to make an IIOP connection back to the server.

Note

The Domino Driver for JDBC, which enables a Java application to access information stored in Domino databases, is currently not available on the AS/400 system. However, it can be used in a Java application or agent running in a Notes client workstation.

5.4.5 DECS

Domino Enterprise Connectivity Services (DECS) is part of the Lotus Domino for AS/400 product. DECS is a Domino server task that allows application developers to link their Lotus Domino databases to relational databases and access data from them in real time. DECS works by capturing certain Lotus Domino database events on the server, such as opening a form and triggering a predefined action on the relational database. DECS was added to Domino R4.6.3 and is still available in Domino R5. There is no additional charge for its use.

A DECS real-time activity defines the way you want your Domino application to interact with your enterprise data. You administer DECS real-time activities through Activity documents. Each document describes one activity by:

- Associating a connection with a form in a Domino database
- Mapping keys and data fields between the Domino form and the enterprise source
- Monitoring an event, such as Create, Open, Update, or Delete

When a Notes client or Web browser client accesses data through a form using a monitored event, DECS intercepts the Domino form processing. DECS processes the associated data on the enterprise source and returns any results to Domino. Domino then interacts with the client in the usual manner.

Using DECS administration, you identify the fields in a Domino application form that map to corresponding entities in an external database. The visual tool provides an application wizard and online documentation to assist the application developer in defining external data source connections to Lotus Domino Connector sources, including relational databases, ERP, and transactional processing systems. Document events that occur with this form, such as open or update, automatically query or update external connector data.

When creating a DECS RealTime activity, several items are required to provide real-time data access from a single Notes form. Each DECS RealTime activity monitors a specific Domino database and requires a Notes form to define the metadata. A single external data source definition indicates the data source to connect to and the metadata to use. Key and field mappings are also required. Several DECS RealTime activities can monitor different databases, a single database or even a single form. Therefore, a single document can be populated in real time with data from multiple external databases, using a DECS RealTime activity for each of the various backend data sources.

The following steps summarize how to use DECS:

1. Build or modify the Domino application to present the external data.
2. Create a connection document for the external data.
3. Create a RealTime Activity between the Domino application and external data.

5.4.5.1 Where to use DECS

Using DECS, you can create Domino applications that access enterprise data in real time without programming.

DECS is the perfect tool to read data in real-time from DB2 UDB for AS/400 or an ERP system on the AS/400 system, while offering high performance in this process of retrieving data.

It can be used for adding, updating, or deleting records in the database. However, it should not be used in a transactional environment. Other tools, such as IBM WebSphere, should be considered when the e-business application is transaction-oriented.

DECS can run simultaneously in several Domino partitioned servers on the same AS/400 system.

Several aspects need to be considered to use DECS in an optimal way:

- Take care in maintaining the keys used to relate the Domino data (in the Domino database) and the external data on the AS/400 system (DB2 UDB for AS/400 or ERP system) consistently.

  For example, DECS does not work if the key used in Domino does not exist in the accessed DB2 UDB for AS/400 table.

  New and deleted keys are not seen in DECS, if they are created or deleted by other AS/400 programs outside Domino.

  Lotus Enterprise Integrator, MQ Series, or additional logic in the application (for example in an agent) should be considered to update the data in both places simultaneously or in a very short time frame, as soon as the application requires it.

- DECS activities need to be started so the Domino application can display or update AS/400 data. We recommend that you start the activities automatically when the DECS task starts, or on schedule (new with Domino for AS/400 Release 5.0.2), and to monitor DECS usage.
- Today, there is no ability to have external data available for use in a view.
- As the user profile used to access AS/400 data is set in the DECS connection document, all updates are made under the same user profile. Several connections can be defined to use the same data source, but a different user profile.

  The Connection Broker MetaConnector (planned for Release 5.0.3 and already available in Lotus Enterprise Integrator) allows data to be directed to and from different connections, based on contact information that is supplied with each row of data. Using this MetaConnector within an Activity form adds in support for enduser authentication to external-connector data for subsequent data write or update operations to connector-defined data.

5.4.6 Lotus Enterprise Integrator (LEI)

Lotus Enterprise Integrator (LEI) is a separately acquired Lotus product for the Domino server that is used for exchanging data between data sources either on schedule or on demand. Data sources can be DB2 UDB for AS/400 tables, Lotus Domino databases, flat files stored in the AS/400 Integrated File System, Zmerge defined text files, and ERP applications for which a Lotus Domino connector is available on the AS/400 system.

The activities that LEI can perform include data transfer, replication, and the execution of LotusScript and Java. It is also capable of real-time data access. Activities may be scheduled, on-demand, or triggered by an event.

LEI is designed to allow moving enterprise data between a relational database, or a flat file, and a Domino database. Although the server is administered by a Lotus Domino application, LEI supports data transfers and replication between non-Domino data sources. Therefore, system managers can adopt a single enterprise LEI system infrastructure to manage interaction between an increasing number of data sources. However, LEI can run only in one partitioned Domino server on the AS/400 system.

LEI can be administered through a Domino R4.6 server or later, and can move data from any Domino server that you administer LEI from the LEI Administrator (leiadm.nsf) Domino application.

5.4.6.1 Where to use LEI

LEI is designed to perform high-volume, high-speed transfers and replications. Data is made available (copied) directly in Domino applications that can be accessed locally by Lotus Notes and Web browser users.

One of the main advantages that LEI provides is the ability to replicate changes that occur in the DB2 UDB for AS/400 data, or the ERP system data, to Domino databases. Those changes are naturally replicated to other Domino servers using the Domino replication which, for example, copies only the changed records instead of copying a whole relational table.

Both Lotus Notes and Web browser users can access the data on any Domino server where data has been replicated. A Notes user can replicate the Domino database on their workstation.

Conversely, data originating on the Web or in Notes clients can also be captured in a Domino application and replicated or transferred to your DB2 for AS/400 tables or ERP systems with LEI.

Figure 50 illustrates this advantage.

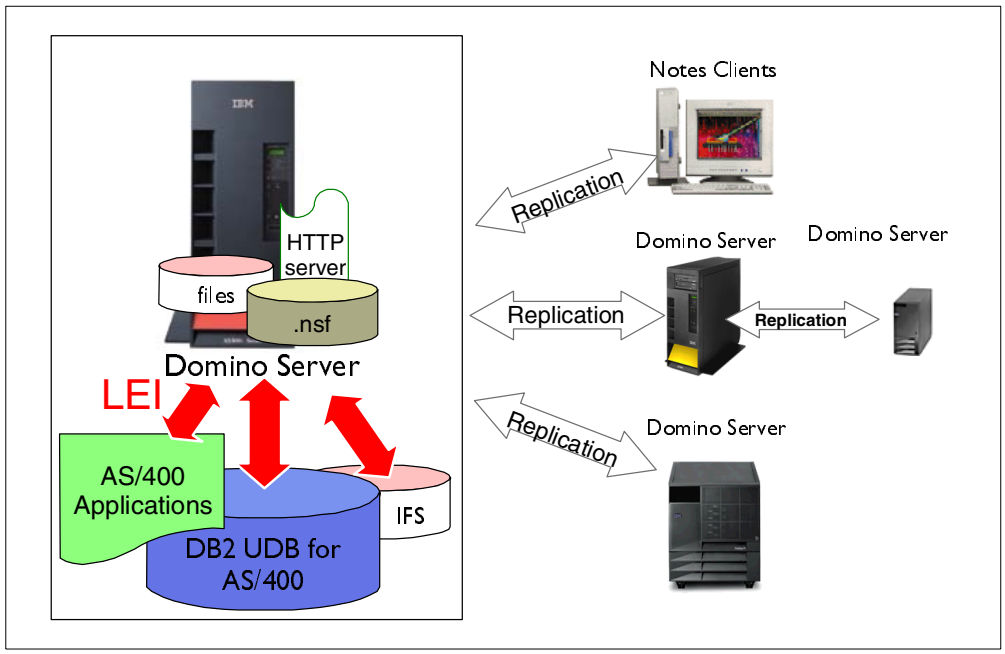


Figure 50. Leveraging Domino replication with LEI

LEI provides other benefits, such as:

- A wide variety of predefined transfer operations
- Productivity (programming is not required)
- “Point and click” server-to-server data processing
- Performance of native connectivity
- Modular architecture with a growing connector set and data manipulators
- High scalability on the AS/400 system

5.4.7 Reference

For more detailed information on Domino for AS/400 enterprise integration tools, refer to the redbooks *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345, and *Lotus Domino 5.0 Enterprise Integration: Architecture and Products*, SG24-5593.

The Enterprise Integration Web page is also a very rich source for information, downloads, white papers, future downloads, discussion databases, and patches. You can find it at: `http://www.eicentral.lotus.com`

5.5 Performance considerations

When talking about performance, people often have different perceptions of what performance is. Here are two ways to look at performance:

- Time aspect—Responsiveness:

  Performance is a measure of time spent on an individual operation or a job.

- Capacity aspect—Throughput:

  Capacity or throughput can be another measure of performance. Performance can be measured in terms of transactions per hour. It can be measured in terms of data transfer rates or resource utilization, such as CPU utilization or disk storage utilization.

The responsiveness and throughput you can reach in your Domino application is a product of:

- Application design and implementation
- Potential interaction with external systems
- Domino Server configuration
- Network configuration
- Server and client operating systems configurations
- Server and client hardware configurations

Often the main factor in a long response time or missing throughput is found in some of the layers below the application.

The number of concurrent users in your application, and what they are doing, will obviously affect performance.

These are application factors that need extra attention to optimize performance:

- Application factors affecting server load:
  - Indexing of views
  - Running of agents
- Application factors affecting user response time:
  - Open database
  - Open view
  - Open document, recalculate document
  - Perform lookup against other database
  - Controlling what is being cached on the server and the client

The first step in examining or building a Web application is designing the architecture. How many pages, views, and databases are involved? Many of the possible answers revolve around a couple of major themes:

- Indexing views takes time and processing power.
- Cached data is served faster than non-cached data.

Domino provides many ways to lay out pages, using both HTML and Domino database constructs, such as forms and fields.

Dynamic pages in Domino are usually produced using forms. Forms provide the ability to programmatically calculate content at many different levels, right down to individual paragraphs and fields. Focus on ways to design forms for maximum performance. The Lotus Web site (located at `http://www.notes.net`) provides you with useful hints and tips on application development.

Remember that the more calculation that is required (through formulas, subforms, and so on) the slower the page will be displayed. If the redirection URL references a database on the same Domino server, you can avoid the extra trip back to the server (and therefore increase response time) by using the double brackets "[[ ]]" around the URL while developing the form.

When you design views into a Web application, keep in mind these two basic principles:

- Indexing views takes time and processing power.
- Smaller views are faster views.

Some changes in the database properties allow you to enhance the overall performance of your e-business applications. With the ODS format, new with Domino Release 5.0, you have the flexibility of changing a number of database properties that can help to improve database and view performance. Table 16 describes these properties.

Table 16. Database properties to enhance performance

Property	Recommendation
Unread marks	In Release 5.0 you can specify whether to maintain unread marks in a Domino database. This was not an option in the past. They were always maintained for each database, regardless of whether they were needed. Maintaining unread marks requires system resources and can significantly slow database performance. The recommendation is to disable this function where possible. This can be done by selecting the database property **Don't maintain unread marks**.
Bitmap optimization	This new table property allows you to update views more efficiently. This property is used to associate tables with the forms that are used by the documents that the tables contain. By selecting document table bitmap optimization, Domino only searches the tables associated with the forms used by documents in the view that you are updating. This significantly improves the performance of view updates, especially when updating small views in large databases. To enable this feature, select **Document table bitmap optimization** on the database properties. **Note**: Whether you select this option, you need to compact the database for the setting to take effect.

Property	Recommendation
$UpdateBy field	The $UpdateBy field contains the name of the user or the server that is associated with each document editing session for a document. The size of this field can grow to a very large size, consuming disk space and having an impact on replication and view update times. To keep this field to a more manageable size and to limit the performance impact, you can select the database property **Limit entries in $UpdateBy fields** and specify the number of entries that this field can contain.
$Revisions field	This is another field that is associated with the editing history of a document. The $Revisions field stores the date and time of each document editing session. Domino uses this field to resolve replication conflicts. The number of edit sessions that are stored in this field by default is 500, which not only consumes disk space, but also slows view updates and replications. The number of edit sessions can be changed by selecting the database property **Limit entries in $Revisions fields** and specify a lower number of edit sessions that can be kept. A suggested limit is 10 entries.
Specialized response hierarchy	If you have views that use `@functions`, `@AllChildren`, and `@AllDescendants` in their selection and replications formulas, you can gain a performance benefit by deselecting this database property. This property can be deselected by selecting the database property **Don't support specialized response hierarchy**. **Note**: Whether this property is selected, you need to compact the database for the change to take affect.
Overwrite free space	To maintain database security, Domino overwrites the deleted data on disk with a pattern. This pattern prevents unauthorized users from using utilities to access the data. Writing this pattern causes extra disk I/O and can affect database performance. If you know your data is already secure, then you may not need to have this overwriting occur. To deselect this property, select **Don't overwrite free space**. Again, this can cause a security risk, so you need to ensure that you will not have any security exposures before you deselect this option to gain the performance benefit.

Formulas, LotusScript, and Java represent just a few of the ways you can program your Web applications using Domino. Also, the technique you use to integrate DB2 UDB for AS/400 data will have a performance impact. Evaluate the use of DECS versus LotusScript, for example.

A solid designer experience is a key asset. For performance hints and tips on Web forms, views, and application design, refer to the Lotus white paper *Maximizing Application and Server Performance in Domino* (in PDF format) at:
`http://www.as400.ibm.com/developer/domino/perform/maxperform.pdf`

For help on evaluating this aspect, use the sizing information available at:
`http://www.as400.ibm.com/domino/D4szintro.htm`

For more detailed information on Domino applications performance and how to improve it, see IBM redbook *Performance Considerations for Domino Applications*, SG24-5602.

Chapter 6. Domino development tools

In this chapter, we introduce some Domino development tools. We show you the functions of these tools and how to install these tools.

6.1 Domino Designer

In Version 4.x, Lotus provided three client programs. These clients were:

- Notes Mail
- Notes Desktop
- Notes

Users wanting to develop databases were required to use the Notes client, which included both the Notes client and a development environment. In Release 5, the development client is now a separate integrated development environment (IDE) and the name has changed to Domino Designer. The goal of Domino Designer R5.0 is to support rapid development of applications that incorporate enterprise data and streamline business processes for the Domino server.

Figure 51 shows the Domino Designer R5.0 user interface.

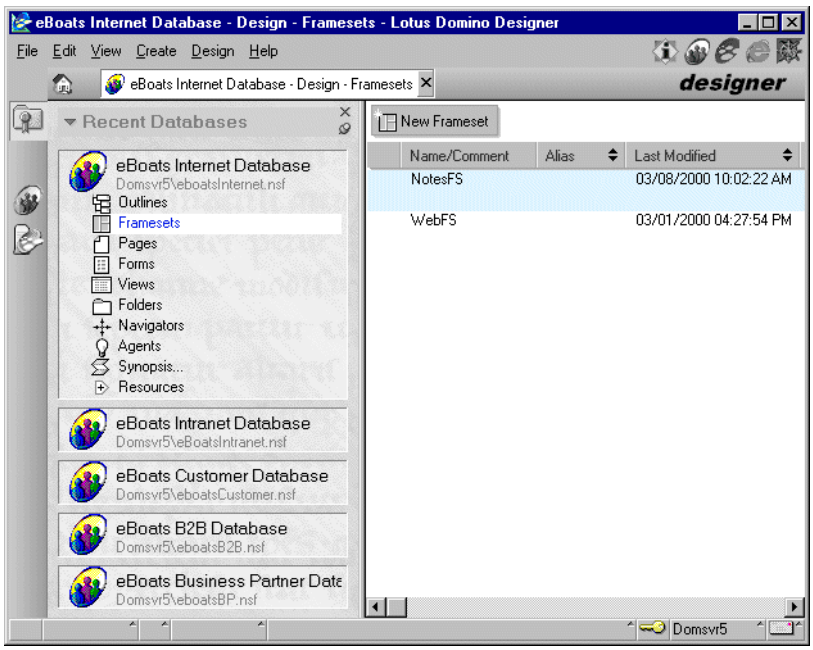


Figure 51. Domino Designer R5.0

6.1.1 Overview of the new features in Domino Designer R5.0

The new features in Domino Designer R5.0 include:

- **Programmability**: Java and JavaScript support are both available within Designer. This allows you to create and compile Java agents, and edit scripts and formulas, all from within Designer.

  Domino R5.0 uses an architecture called Common Object Request Broker Architecture (CORBA). CORBA serves as middleware for a distributed computing environment where remote clients can invoke methods on remote

APIs residing on other computers. CORBA/IIOP support enables Domino developers to create applets that can be downloaded to the client and can be remotely invoked in Domino services, for example, to initiate a workflow process.

You can import full-fidelity HTML files (including applets and animated GIFs) from anywhere on the Web and display them with Designer. HTML 4.0 is fully supported.

- **Outlines**: The new *Outline* design element allows easy navigation through an application and its elements. Outlines provide you with a display of the hierarchical tree structure of links and elements in your site or application. An outline is basically a site map.
- **Pages**: The *page designer* allows you to work within the familiar page environment of the Web. A page can contain anything a form contains, except fields, subforms, and layout regions. With the addition of pages and improved layout and image support, you no longer have to program Web pages in native HTML. You can create the same great-looking Web pages in Designer.
- **Framesets**: Frames allow you to have a navigation panel that is consistent throughout your site. R5.0 includes framesets as a new design element, so you can easily create multi-pane interfaces in your applications. Featured inside the Frameset component is a wizard that allows you to start designing frames quickly.
- **Resources**: *Resources* are shared objects available for your use within a database. They include design elements from previous releases, such as subforms and script libraries, and new design elements, such as images, applets, and shared actions. You develop and maintain the shared resource once and use it throughout the application.
- **Images and text**: In R5.0, you gain complete control over the way that text and graphics behave. You can wrap text around images and tables. You can anchor objects to a position in a paragraph, position paragraphs using nested tables, and position graphic elements in the background.

  An *imagemap* is a graphic that you place programmable hotspots on. These hotspots perform an action when a user clicks them. You can now place imagemaps on documents or forms, so you do not have to use a Navigator to draw multiple hotspots on a picture in a document.
- **Tables**: There is increased *table* support in R5.0. You can use nested tables. You can place text in cells and anchor it. You can create tabbed tables, place sections into tables, and collapse tables. Timed tables allow you to create the illusion of an animated GIF by displaying a different table row, containing a graphic, every specified number of seconds.
- **Design synopsis**: Design synopsis is a redesigned version of the reporting tool that can help you analyze your applications by providing you with a complete, structured list of the forms, pages, and scripts that make up an application.

6.1.2 Domino Designer R5.0 system requirements

The requirements for using Domino Designer R5.0 are:

- **Supported platforms**:
  - Microsoft Windows 95
  - Microsoft Windows 98
  - Microsoft Windows NT 4.0 Workstation
  - Apple Mac PowerPC 7.6 and 8.1
- **Memory requirements**:
  - Microsoft Windows 95 and 98: Minimum 8 MB; 32 MB or more recommended.
  - Microsoft Windows NT 4.0: Minimum 16 MB; 32 MB or more recommended.
- **Disk Space**: Minimum 70 MB; 236 MB or more recommended.
- **Protocols**:
  - TCP/IP, SPX, NetBIOS/NetBEUI, VINES, ISDN, and X.PC supported on all platforms.
  - SPXII supported on selected platforms.

Note

TCP/IP is the only supported protocol to connect to a Domino for AS/400 server.

6.2 WebSphere Studio

IBM WebSphere Studio is a very rich Web development team environment available for organizing and managing Web development projects. Studio offers unparalleled ease-of-use features, making it the most complete Web development product available today for developing and deploying interactive Web sites for the IBM WebSphere Application Server. Best-of-breed tools let you easily import an existing HTML Web site and start adding data-driven JavaServer Pages (JSPs). Studio provides everything you need to support your Web application development needs, from end-to-end Web site development to remote server-side logic debugging to deployment of your Web-based applications.

Studio delivers a complete set of Web development tools with clear advantages of performance, portability, and lower maintenance over other approaches such as CGI and ASP. Studio combines graphical development wizards with tools for Web site design and Java development, with features including:

- A workbench environment that lets Web development teams organize and manage Web development projects. This environment can be extended with source control management (SCM) tools. Of particular interest is the ability to use Studio with the Lotus Domino Web Content Library, which, in addition to versioning support, allows integration with Domino workflow.
- An industry-leading, easy-to-use, integrated visual page designer for JSPs, HTML, and DHTML that accelerates Web page development.

- An integrated remote debugger for easy remote debug of server-side scripts and logic (including JSP components, servlets, JavaBean components, and more). The remote debugger requires the Standard or Advanced edition of IBM WebSphere Application Server.
- Integrated wizards to help developers create dynamic interactive Web pages. Wizards generate JSPs, JavaBeans, SQL statements, and servlets.
- Tighter integration between IBM VisualAge for Java Professional Edition, V3.0 and Studio, making it easy for teams to communicate and work together to develop Web-based, e-business applications.
- An integrated applet designer based on the NetObjects BeanBuilder technology.
- NetObjects ScriptBuilder allows for easier script editing of Extensible Markup Language (XML) and Wireless Markup Language (WML).
- An integrated Web art designer that lets you create masthead images, buttons, and other graphics.
- An integrated animated GIF designer that makes it easier to create animated GIFs.
- A development copy of IBM WebSphere Application Server for building and testing Web applications.

There are two main reasons why you should consider WebSphere Studio as a development tool in a Domino for AS/400 environment:

- Some features, such as integrated visual page designer for HTML, the applet designer, NetObjects ScriptBuilder, integrated animated GIF designer and integrated Web art designer, are very useful to create a Web page. The Web page can be shown by the Domino HTTP server.
- WebSphere Studio can be integrated very well with WebSphere Application Server.

6.2.1 System requirements

The requirements for using WebSphere Studio V3.0 are:

- **Minimum hardware requirements**
  - Any Intel Pentium class PC running Microsoft Windows NT server, Version 4.0 with Service Pack 3, Windows 95, or Windows 98
  - 180 MB of free disk space for installation
  - CD-ROM drive
  - 32 MB of memory
  - VGA, or better video adapter, configured for at least 256 colors
- **Minimum software requirements**
  - Windows 95, Windows 98, Windows 2000, or Windows NT, Version 4.0 with Service Pack 3
  - Microsoft Internet Explorer, Version 4.0 or higher

6.2.2 Installing WebSphere Studio Version 3.0

Installing the WebSphere Studio Version 3.0 is easy. This section shows you the step-by-step installation procedure on Windows NT 4.0. To install the WebSphere Studio Version 3.0, you should have the WebSphere Studio CD in your CD drive. Then, follow these steps:

1. Try to locate the file “setup.exe” at the CD root directory. Double-click the file on Explore. The first display that appears is shown in Figure 52.

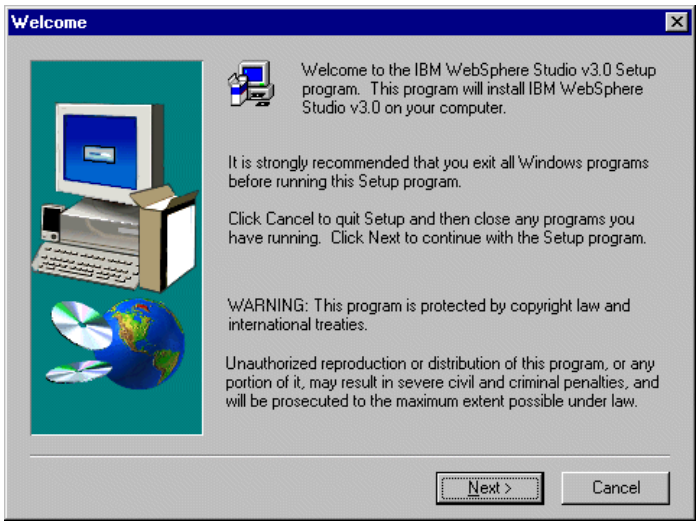


Figure 52. Setting up WebSphere Studio V3.0

2. Click **Next**, read the Software License Agreement, and click **Yes**. Then, you have to select the installation directory. See Figure 53.

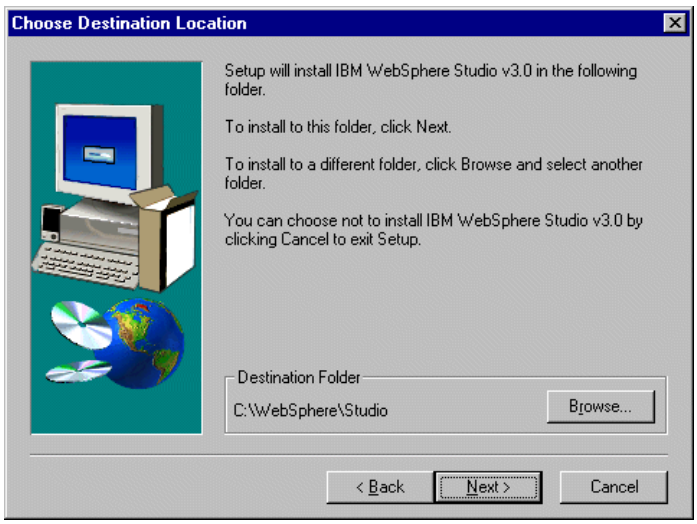


Figure 53. Setting up WebSphere Studio: Choose Destination Location

3. Click **Browse** to select your directory. Make sure the space in your destination drive is enough. The hard disk requirement are described at 6.2.1, “System requirements” on page 136. On the next display, select the components to install. Select both of the components. See Figure 54 on page 138.

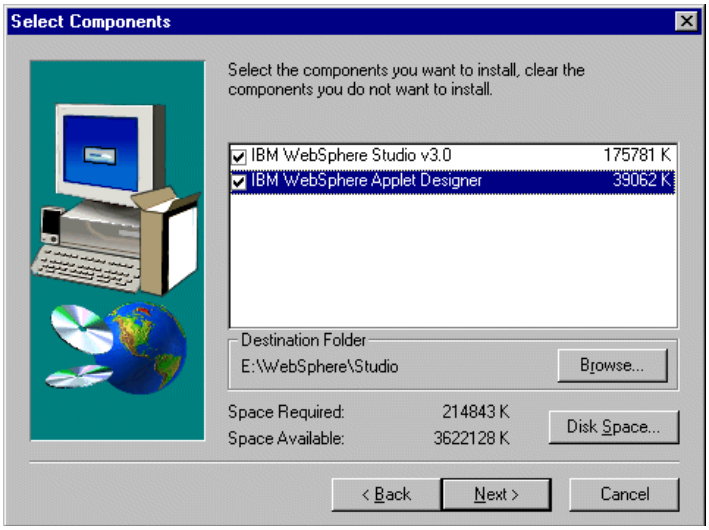


Figure 54. Setting up WebSphere Studio V3.0: Select Components

4. Click **Next** to start the installation. It will take several minutes to finish. When it is finished, you should restart your machine.

6.2.2.1 Starting WebSphere Studio V3.0 the first time

To start WebSphere Studio V3.0, follow these steps:

1. From the Windows NT Start menu, click **Program->IBM WebSphere->Studio 3.0->IBM WebSphere Studio V3.0** to start WebSphere Studio V3.0. The first time you start, you are required to create a new project or open a existing project.
2. Choose **Create a new project**, and then specify a project name. A directory with the same name as the project name is created under the \WebSphere\Studio\projects\ directory. A project file that has a suffix of .wao is created in the new directory. As shown in Figure 55, select **Create a new project**.

Figure 55. Start WebSphere at the first time

Figure 56 show how to specify a new project information. When you type the project name, the directory is entered automatically.

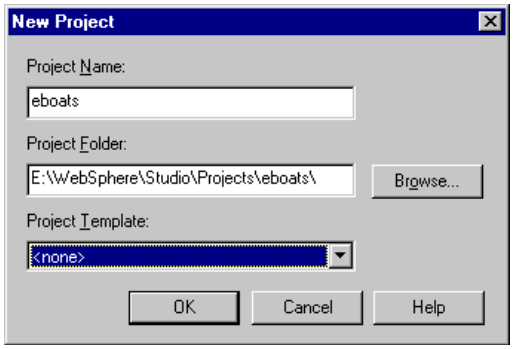


Figure 56. New project for WebSphere Studio

3. Click **OK**. Your installation is finished. Figure 57 shows the WebSphere Studio workbench.

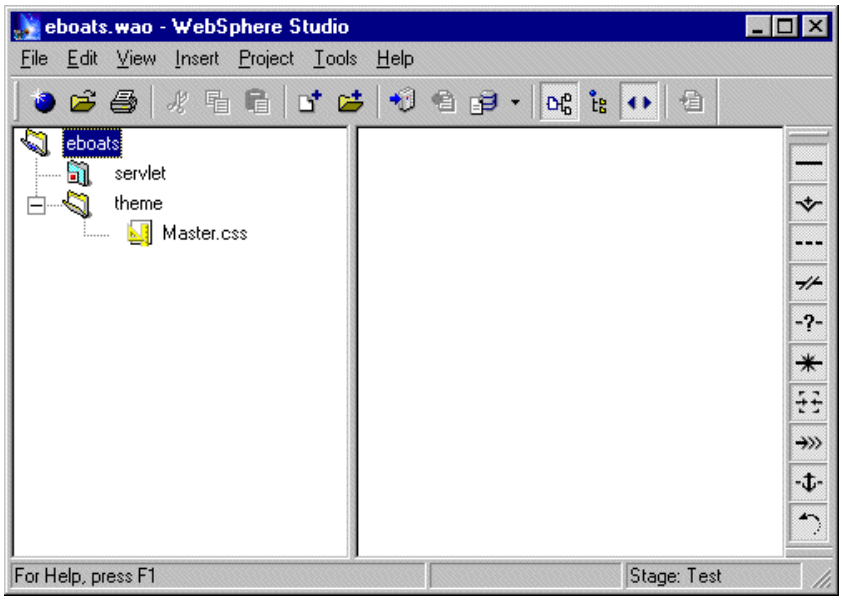


Figure 57. WebSphere Studio workbench interface

6.2.2.2 Setting up the environment

To access the DB2 UDB for AS/400, you can use AS/400 Java Tool Box. There is a jt400.zip file and a jt400.jar file included in the AS/400 Java Tool Box. You can find these two files in AS/400 Integrated File System. The directory is /QIBM/ProdData/Http/Public/jt400/lib. You can copy them to your local directory. In our example, the files are located at e:\jt400\lib\. You should add the file name to CLASSPATH of Windows NT. Follow these steps:

1. Go to the Control Panel and select the **System** icon.
2. Click the **Environment** tab.
3. Modify CLASSPASS in the **User Variables for Administrator** list. See Figure 58 on page 140 for an example.

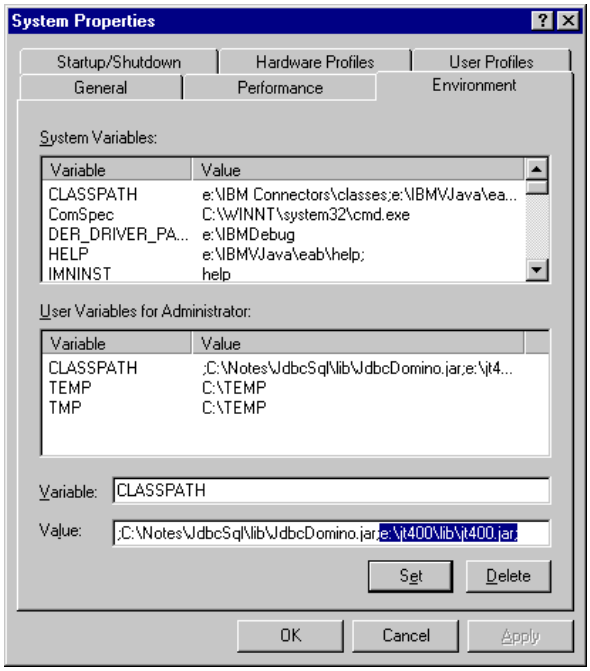


Figure 58. CLASS PATH setting for administrator

4. After setting the CLASSPASS, you can try to verify the DB2 UDB for AS/400 connection. Click the project name in the left panel of the Studio workbench.
5. Click **Tools->Wizards->SQL Wizard**. The SQL Wizard panel pops up. See Figure 59 for an example.

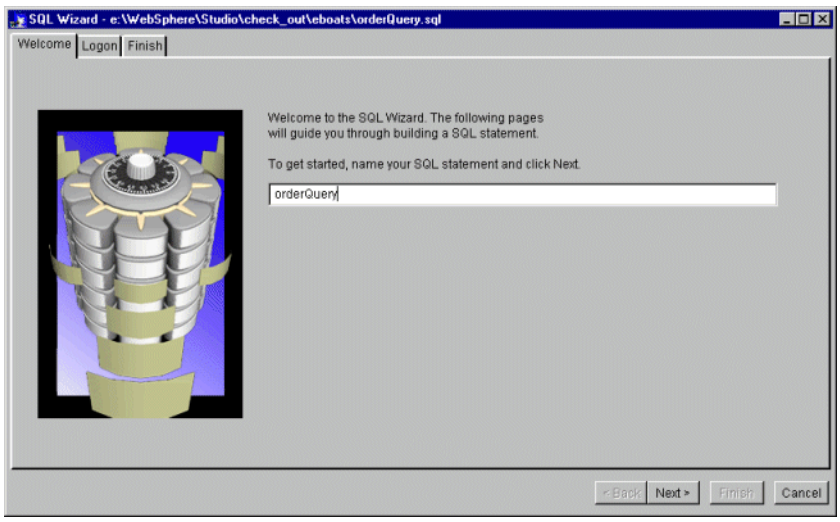


Figure 59. New query file

6. Click the **Next** button and fill in the connection parameters. See Figure 60.

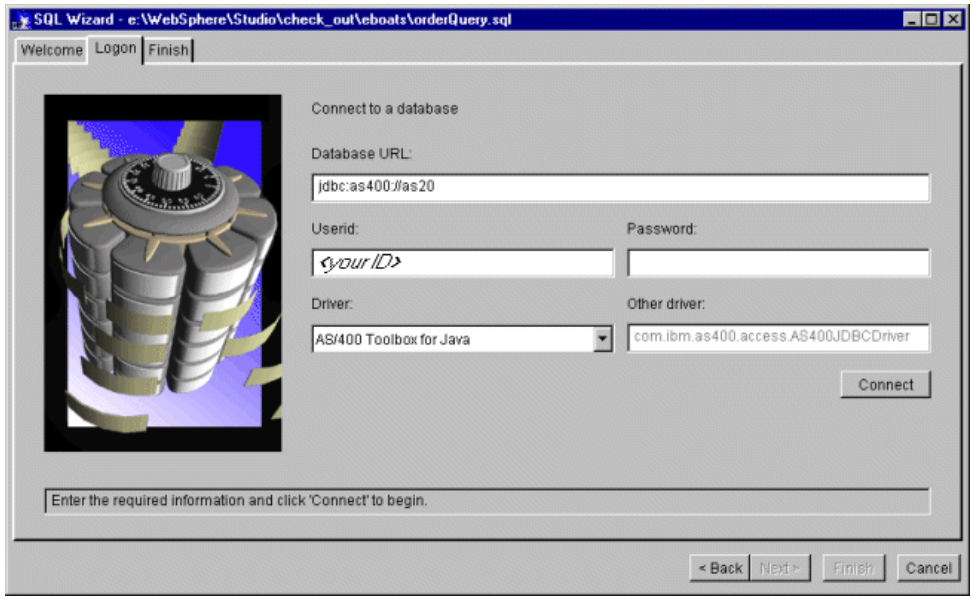


Figure 60. Connection parameters

Click **Connect** to connect SQL Wizard to the AS/400 system. If connected, the Library list of the AS/400 server is listed. Follow the wizard step-by-step to finish the definition of SQL statement. A file with suffix .sql is created in your workbench. The final result is shown in Figure 61.

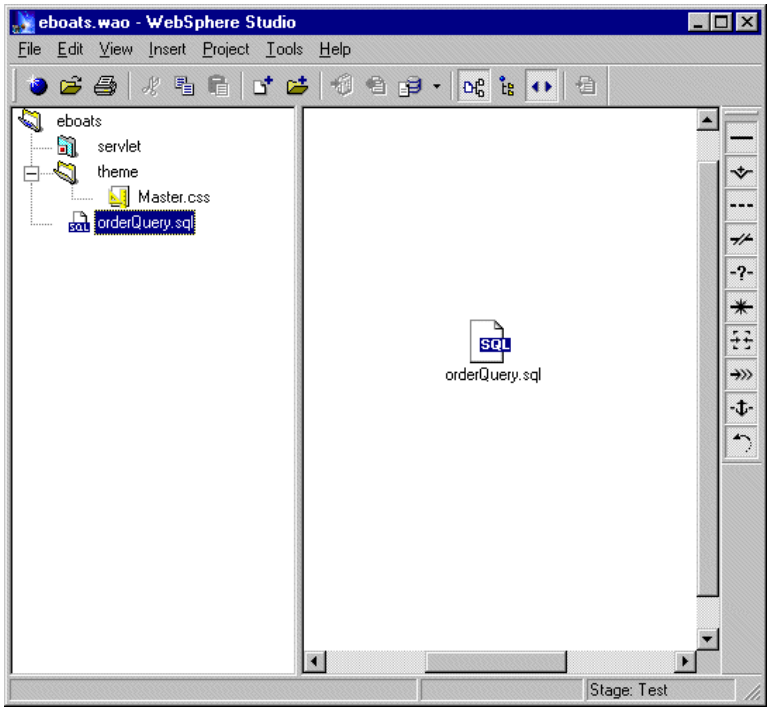


Figure 61. Result of the SQL Wizard

Congratulations! You have now finished the setup.

6.3 VisualAge for Java

VisualAge for Java, Version 3.0, includes new features and performance improvements that make it easier than ever to create scalable, hard-working e-business applications. Tighter integration with IBM WebSphere Application Server, WebSphere Studio, and DB2 Universal Database speed development time and improve productivity, while providing easier, secure access to enterprise data:

- VisualAge for Java is closely linked with WebSphere Application Server, which enables developers to remotely test and debug applications, cutting development time.
- Tight integration with WebSphere Studio makes it easy for teams to communicate and work together to develop Web-based, e-business applications.
- A new Stored Procedure Builder helps developers create stored procedures for IBM DB2 Universal Database, so applications can benefit from DB2's rock-solid security features.
- New SQLJ support speeds data access response time and simplifies SQL coding.
- Data access beans allow application developers to easily access JDBC-enabled relational databases from a single Java application.
- The Enterprise Access Builder (EAB) for enhanced connectivity to enterprise application servers has been simplified and includes new support for IMS, MQSeries, Notes, Host on Demand, and EJB support for SAP R/3. The EAB extends existing enterprise systems to the Web.
- A standards-based XML Metadata Interchange (XMI) bridge between VisualAge for Java and Rational Rose, providing complete generation of Java code from Rational Rose models and Rational Rose models from VisualAge for Java.

The reason we are using VisualAge for Java in Domino application development is:

- VisualAge for Java is an integrated Java development environment, including coding, debugging, testing and publishing functions.
- Using VisualAge for Java, you can create Java agents, applets, and servlets for Domino more easily.

6.3.1 VisualAge for Java, Enterprise Edition, Version 3.0

The following components are included with VisualAge for Java, Enterprise Edition:

- **Integrated Development Environment (IDE)**: This is a set of windows that provide the user with access to development tools. The primary windows are the Workbench, Log, Console, Debugger, and Repository Explorer. You can use it to create Java applets, which run in Web browsers, and standalone Java applications.
- **Visual Composition Editor (VCE)**: You can use this tool to create graphical user interfaces from prefabricated beans and to define relationships (called connections) between beans. This process of creating object-oriented

programs by manipulating graphical representations of components is called *visual programming*.

- **Team Programming support**: This product provides an integrated team development environment that is based on a shared source code repository
- **Builders**: The following builders are available:
  - The Enterprise Access Builder (EAB) for Transactions helps developers to quickly and easily extend existing applications to the Web and e-business. It consists of frameworks and tools that allow you to access the function and data assets of your Enterprise Information Systems (EIS). It provides a standard access builder interface for connectors that adhere to the IBM Common Connector Framework (CCF). EAB uses e-business connectors that are based on CCF. Connectors are provided for CICS (both ECI and EPI), Encina DE-Light, IMS TOC, MQSeries, Host-on-Demand (HOD), and SAP R/3.
  - The Access Builder for SAP R/3 is a complete toolkit for creating Java applications, applets, and beans that allow access to an SAP R/3 system. The Access Builder for SAP R/3 helps you to retrieve complete meta-information within the R/3 system, keep it locally for multiple R/3 systems, access without an R/3 connection, and search for business objects. It includes a connector for SAP R/3 that generates EJB proxies for business objects.
  - The RMI Access Builder enables you to provide remote access to Java beans in a distributed, client-server environment. You can use it to generate proxy beans and associated classes and interfaces so you can distribute code for remote access, enabling Java-to-Java solutions, as well distribution-specific code that supports the distribution of bean events and integrates with the application-specific code contained in your bean.
  - The C++ Access Builder provides access from Java applets or stand-alone Java applications to services written in C++. You can use it to generate beans and C++ wrappers that let your Java programs access C++.
  - The Domino Access Builder allows you to create Java applications that access databases and services located on a Domino server or on a Notes client. Domino Access Builder provides generic beans, based on the Domino Java classes from Lotus, plus a SmartGuide to create user-defined beans. The generic beans include wrapper classes for databases, forms, views, and other Domino design elements. You can use the SmartGuide to select a database and to configure the code generator, which creates customized beans that match the selected Domino design elements. As well, the Domino Access Builder enables developers to specify which view, field, or columns are represented in the JavaBeans.
- **Data Access**: These beans provided with VisualAge for Java, for developing programs that access relational databases:
  - The DB2 Stored Procedure Builder (SPB) is a tool that enables you to write Java stored procedures to run on a DB2 server.
  - SQLJ support provides you with a standard way to embed SQL statements in Java programs. The SQLJ standard has three components: embedded SQLJ, a translator, and a runtime environment. The translator translates SQLJ files that contain embedded SQLJ to produce .java files and profiles that use the runtime environment to perform SQL operations. The runtime

environment usually performs the SQL operations in JDBC and uses the profile to obtain details about database connections.

  - The Enterprise Access Builder for Persistence (Persistence Builder) enables you to map objects and relationships between objects to information stored in relational databases. It also provides linkages to the mapping between Enterprise JavaBeans and relational data.

- **Distributed Debugger**: This client/server application enables you to detect and diagnose errors in your programs. You can use it to debug Java applications that are developed outside the IDE.
- **Domino AgentRunner**: The Domino AgentRunner helps you build, run, and debug Domino agents in VisualAge for Java. It uses a set of debug classes that access Lotus Notes context information so that you can run and debug an agent in the VisualAge for Java Integrated Development Environment (IDE).
- **EJB Development Environment**: The EJB Development Environment enables you to develop and test Enterprise JavaBeans (EJB) beans and access (adapter) beans for use in your Enterprise applications. In addition, the EJB Development Environment provides all of the necessary run-time support for the IBM WebSphere Application Server. It includes a new incremental consistency checker that ensures that an enterprise bean conforms to the EJB programming specification and indicates whether changes are needed to fix inconsistencies.
- **Enterprise Toolkits**: The Enterprise Toolkit for AS/400 (ET/400) allows you to develop enterprise applications in Java for the AS/400 platform. You can use ET/400 to export Java class and source files to the AS/400 and compile Java code optimized for the AS/400 system and to run and debug AS/400 Java applications from the VisualAge for Java IDE.
- **External SCM Tools**: This is a bridge that allows you to use an external software configuration management (SCM) system from within VisualAge for Java. You can add classes to source control, check classes in and out, and import the most recently checked-in version of a class into VisualAge for Java.
- **IDL Development Environment**: This provides an integrated interface definition language (IDL) and Java development environment that works with a user-specified IDL-to-Java compiler. You can work with both your IDL source and the generated Java code in one convenient browser page to perform such tasks as importing IDL files and creating new IDL groups and IDL objects.
- **JSP/Servlet Development Environment**: This tool enables you to develop and test JavaServer Pages. You can use all of the Java APIs that are available to any Java applet or application. For example, you can use JavaBeans, Enterprise JavaBeans (EJB), and servlets.
- **Migration Assistant**: You can use the Migration Assistant to create a code framework for a JavaBean to replace an ActiveX control. It creates a set of methods, properties, and enums that an equivalent JavaBean needs to achieve the same function as an ActiveX control.
- **Tivoli Connection**: Tivoli Connection helps you create business-critical distributed Java applications that can be managed by Tivoli's enterprise management software. Tivoli Connection allows you to generate Tivoli events and to interface to the Tivoli Enterprise Console. Tivoli provides an application management solution that covers the life cycle of an application, from deployment to monitoring and administration.

- **Tool Integrator**: This is a VisualAge for Java mechanism that lets you integrate Java applications that reside on the file system to launch them from within the IDE.
- **XML Metadata Interchange (XMI) toolkit**: The XML Metadata Interchange (XMI) toolkit for integration with the Rational Rose modeling tool. VisualAge for Java converts the Rose model (.mdl files) into an XMI format. Since VisualAge understands the parsing of XMI (the XML data type definition for UML), the generation of the Java code is done inside the development environment. A comparison tool reconciles what is contained in Rose with the implementation in VisualAge for Java, and this facilitates the rapid transformation of business models.

6.3.1.1 New features

The following features are available in Version 3.0:

- WebSphere Test Environment
- JSP/Servlet Development Environment
- Stored Procedure Builder
- Support for SQLJ
- EJB Development Environment
- Domino Access Builder

6.3.1.2 Enhancements to existing features

VisualAge for Java, Enterprise Edition, Version 3.0 also offers the following enhancements:

- Additional data access beans for creating, reading, updating, and deleting data and for accessing stored procedures
- Customizable browsing and enhanced code assist features in the IDE, including macro support
- Contextual help for Java keywords and program elements
- Distributed debugger for Java applications that run across different environments
- Enhanced integrated debugger accessing stored procedures
- Performance improvements to code generated by the Visual Composition Editor
- Other enhancements to the Visual Composition Editor, including improved property editor support and Null-to-GridBag layout converter
- JDK updated to Version 1.1.7A, featuring euro currency support
- JFC updated to Version 1.0.3
- Extensions to the Tool Integrator API, allowing for remote access, so that tools can now run outside the IDE
- An incremental consistency checker that confirms that enterprise beans conform to EJB programming specifications
- Support for enterprise beans from the Access Builder for SAP R/3
- In the Enterprise Access Builder, new e-business connector beans that support the Common Connector Framework (Host-on-Demand, IMS, MQSeries, and SAP connectors

6.3.2 System requirements

This edition of VisualAge for Java, Version 3.0, has the following hardware and software prerequisites:

- **Minimum hardware requirements**
  - TCP/IP installed and configured
  - Pentium processor or higher recommended (minimum processor speed 90 Mhz)
  - A mouse or pointing device
  - SVGA (800x600) display or higher
  - 64 MB RAM minimum (96 MB recommended)
  - If you are migrating from Version 2.0 of VisualAge for Java, your migrated repository will increase in size to include the repository shipped with Version 3.0. You can expect an increase in the size of the repository of at least 35 MB for a basic installation.
- **Minimum software requirements**
  - Windows 95, Windows 98, or Windows NT 4.0 with Service Pack 3 or Service Pack 4. For customers who want to use the euro currency symbol on Windows, a free update is available from Microsoft from the following Web address: `http://microsoft.com/windows/euro.asp`

    This patch is not required if you are using Windows NT 4.0 with SP4 applied.
  - Frames-capable Web browser such as Netscape Navigator 4.04 or higher, or Microsoft Internet Explorer 4.01 or higher. The recommended browser is Netscape Navigator 4.6 or Internet Explorer 5.0.
  - Java Development Kit (JDK) (TM) 1.1.7 for deploying all applications.

6.3.3 Installing of VisualAge for Java, Enterprise Edition, Version 3.0

To install VisualAge for Java, Enterprise Edition, Version 3.0, follow these steps:

1. Insert your CD in the CD drive.
2. Run the setup.exe file in the “/” directory of the CD drive and start the installation. Figure 62 shows the first display that appears.

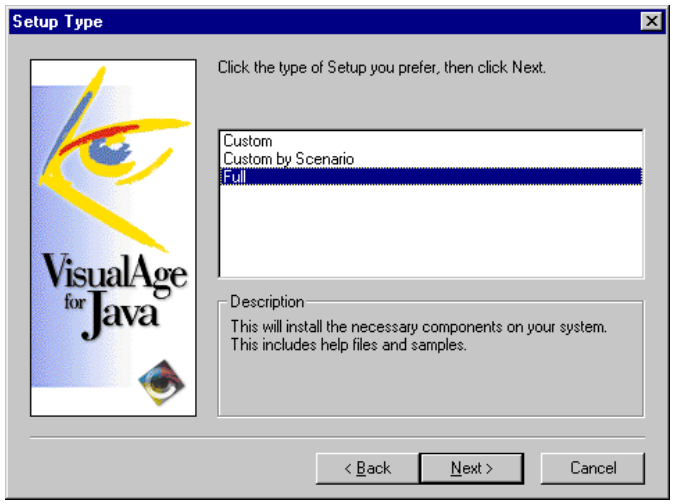


Figure 62. Starting the installation of VisualAge for Java, Enterprise Edition, Version 3.0

3. Select **Full**, and click **Next** to continue. Now you need to select the repository, as shown in Figure 63.

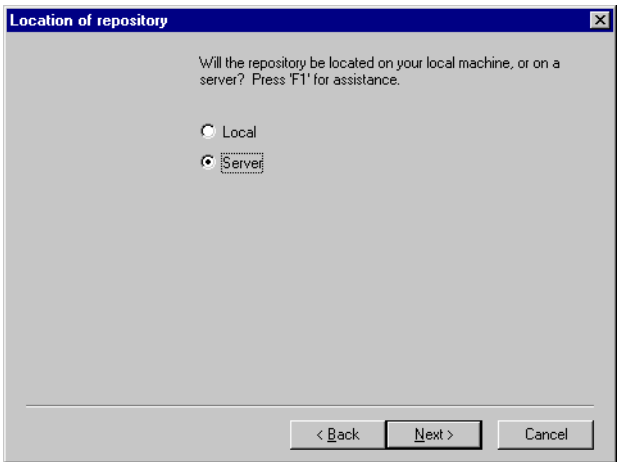


Figure 63. Setting up VisualAge for Java: Selecting the repository

4. VisualAge for Java supports team programming. Team Server for VisualAge for Java should be installed when you choose team programming feature. In our case, Team Server is already installed on one machine, and a server repository is created on the machine. Here you should select **Server** as the repository. Click **Next** to go to next display, as shown in Figure 64 on page 148.

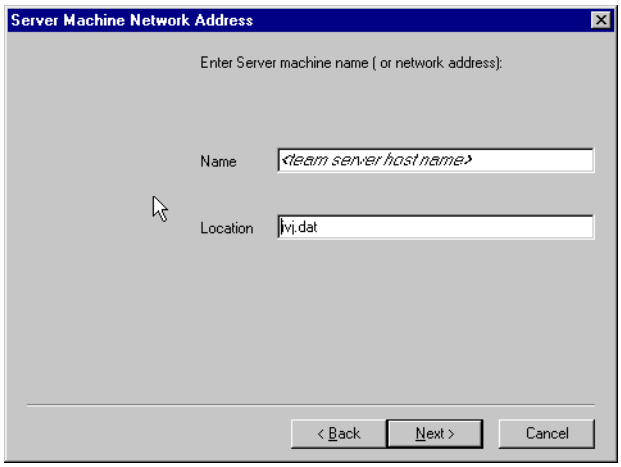


Figure 64. Setting up VisualAge for Java: Locating the server repository

5. Input the server machine name or network address at the Name field. The Location should be kept as default value `ivj.dat`. Click **Next** to go to next display, which is shown in Figure 65.

Figure 65. Setting up VisualAge for Java: Select program folder

6. Click **Next** twice to start the file copy process, which is shown in Figure 66.

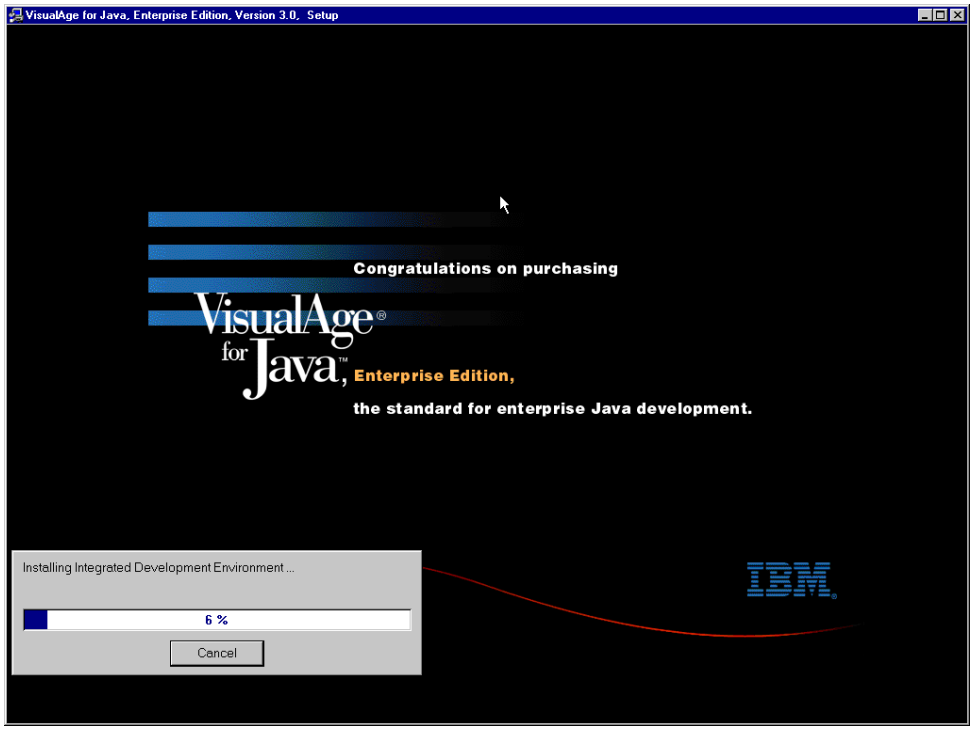


Figure 66. Setting up VisualAge for Java, Enterprise Edition, Version 3.0: Copying the file

When the file copy is finished, restart your machine again. The installation of VisualAge for Java, Enterprise Edition, V3.0 is completed.

6.4 Third-party tools

With the growth of the Internet, companies of all sides are rushing to develop a Web presence. These Web sites can range from static HTML pages describing the company and its product lines to dynamic e-commerce sites whose content can change hourly. This growth has transformed the role of Web developers from obscurity to the corporate “front line”. To be successful in the Internet arena, companies will be required to reduce the time it takes for them to move to the Web and to reduce development costs. This can be accomplished by maximizing productivity by using the best tool for the job, efficiently using current skill sets, and keeping abreast of technological innovation by using the latest and greatest tools.

Developing applications for the Web today has increased in complexity. Web application development teams are growing but not fast enough to keep up with demand. One solution is to empower employees to publish and contribute content. Every company has their share of “power users” who have the technical knowledge to manage content for the Web site that pertains to their area of expertise. The most commonly used tools by these “power users” are NetObjects Fusion and Microsoft FrontPage. These tools are easy to use, require no programming experience, and allow you to create a nice looking Web site. Web sites created with these tools are generally static in nature. Companies that want to maximize the return on investment for the Web sites soon realize that they need to integrate with mission-critical applications and with the information-flow

process in the organization. This requires transforming the sites from static to dynamic and data-driven.

The strength of Domino lies in the fact that end-users can create and manage their documents, its workflow process, e-mail capabilities, enterprise integration, and the ability to develop mission-critical applications.

Integrating Domino with third-party HTML authoring tools can extend the value and functionality of current Web sites. Lotus provides a set of components, called Domino Design Components for Microsoft Front Page and NetObjects Fusion. These components allow a user to add Domino design elements such as views, forms, search, and database links to Web sites. An add-in server task, called the Domino Import Service, allows you to publish a FrontPage or NetObjects Fusion site directly to Domino. The Domino Design Components are free and can be downloaded from: `http://www.lotus.com/Webauthor`

Together, they provide Web developers and site designers access to the enterprise integration, workflow, and collaborative capabilities of Domino.

6.4.1 The Domino design components

The same set of design components are available for both NetObjects Fusion and Microsoft FrontPage. The components are:

- **Domino Site**: Places a Domino database (NSF file) on a Web page.
- **Domino Link**: Links to a Web site on another server, to a Web site on the Internet, or to a Domino database by creating a Domino URL.
- **Domino Form**: Allows you to create a Domino form on a Web page.
- **Domino View**: Allows you to create a Domino view or display an existing Domino view on a Web page.
- **Domino ViewList**: Displays a list of the views in the Domino site.
- **Domino ViewAction**: Provides navigation controls for a Domino view.
- **Domino Search**: Adds Domino search capabilities to a Domino site.

6.5 Summary

You use different tools for different purposes. You can use Domino Designer to create the data Forms and data Views for data presenting and content management. Using Domino Designer, you can build up your Web site very easily, because Domino Designer is easy to learn and easy to use.

With WebSphere studio, you can create HTML files, Servlets, JSPs and JavaBeans automatically, or create applets and animated GIF by using Wizards.

VisualAge for Java is a powerful Java development tool. The Integrated Development Environment (IDE) makes Java development very interesting. You simply need to click your mouse, and a Java applet will be finished in several minutes. And, when the coding is inevitable, VisualAge for Java makes coding more easier. The version control and team management functions of VisualAge for Java is very strong. That is very important in a team development environment. When you try to create an Enterprise JavaBean, the most powerful tool you can use is VisualAge for Java.

The relationship of these three tools is shown in Figure 67.

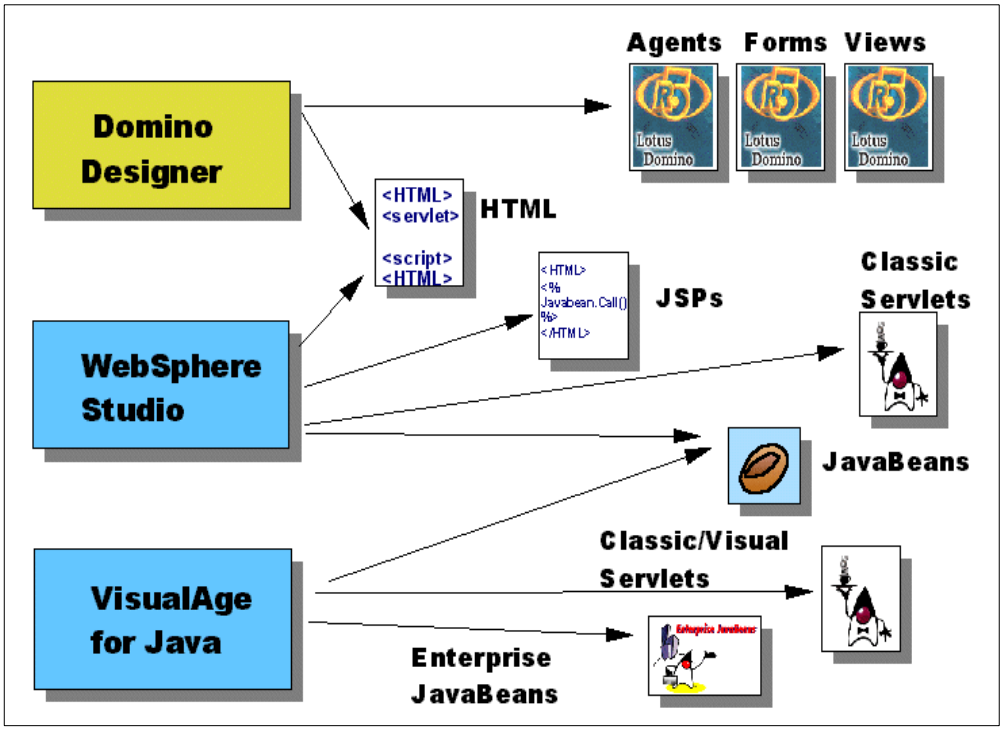


Figure 67. Application development tool's choice

Part 2. Redbook project

This second part covers the following topics:

- Sample project case study
- Sample project development
- Sample project Domino server (and LEI server) setup
- Sample project user management
- Sample project SSL configuration
- Using the sample project application

Chapter 7. Case study

There are five Domino databases and four DB2 UDB for AS/400 tables that were developed to demonstrate the development features discussed in this redbook. This chapter describes the sample application that takes advantage of the advanced features found in Domino R5.0 and the integration with DB2 UDB for AS/400. The databases and tables were developed for a company called eBoats International. A description of eBoats International is provided.

7.1 eBoats International

The sample databases presented in this redbook are from a company called eBoats International. eBoats International is a fictional company that sells luxury sailboats and motorboats. The company, headquartered in Atlanta, Georgia, USA, employs 200 people. eBoats International only sells its custom designed products through selected business partners worldwide. eBoats International maintains a Web site that contains information about its product line and Business Partners as well as a gateway to the company's extranet and intranet sites.

7.2 Application overview

The sample application does not have the level of complexity normally found in mission-critical applications. The database was intentionally kept simple to assist in the description of the desired functionality and to allow the reader to understand the concepts being presented. The application demonstrates the many uses of Domino technology including:

- **Internet**: eBoats Web presence site available to the public
- **Business-to-business/extranet**: Accessible only to registered business partners
- **Intranet**: Accessible only to eBoats International employees

There are five databases that are used to demonstrate all of the functionality described in this redbook. The five databases are:

- eboatsInternet.nsf (Internet)
- eboatsB2B.nsf (extranet)
- eboatsIntranet.nsf (intranet)
- eboatsBP.nsf (business partners)
- eboatsCustomer.nsf (customers)

Figure 68 on page 156 shows how the sample application implements the above uses of Domino.

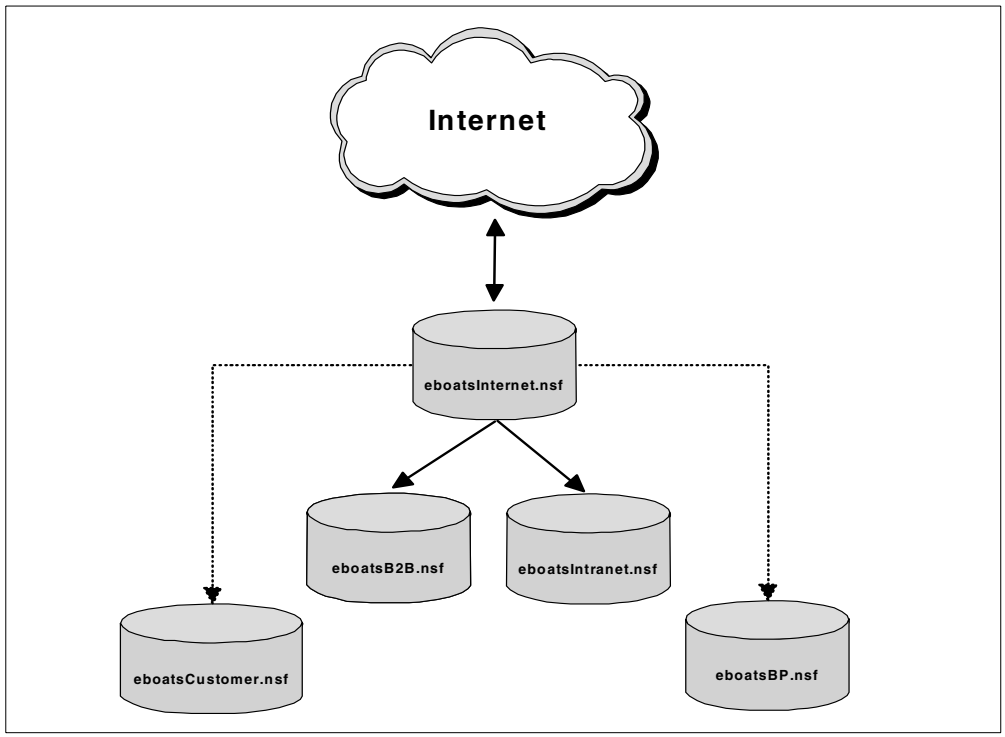


Figure 68. Overview of the features demonstrated in the sample database

7.3 Database descriptions

This section describes the five sample databases. We describe what users can do in each database, and then we summarize the functionality demonstrated.

7.3.1 Internet database overview

This database is the entry point to the eBoats International Internet Web site. Potential customers can view information about the company and the entire product line. If a customer wants more information on a particular product, they can submit an information request online. This form is automatically routed to the nearest Business Partner using Domino's workflow technology. Companies that want to become an eBoats International business partner can submit an application online. Users that want to purchase any of their products can find the name and address of the business partner closest to them.

This database also contains the login point for both the company's extranet site and intranet site. Business partners logon to the extranet site, and eBoats International employees logon to the intranet site.

This database demonstrates the use of:

- JavaScript
- @functions
- SSL
- X.509 certificates
- LEI

Figure 69 shows the technologies used in this database and the relationship with other databases.

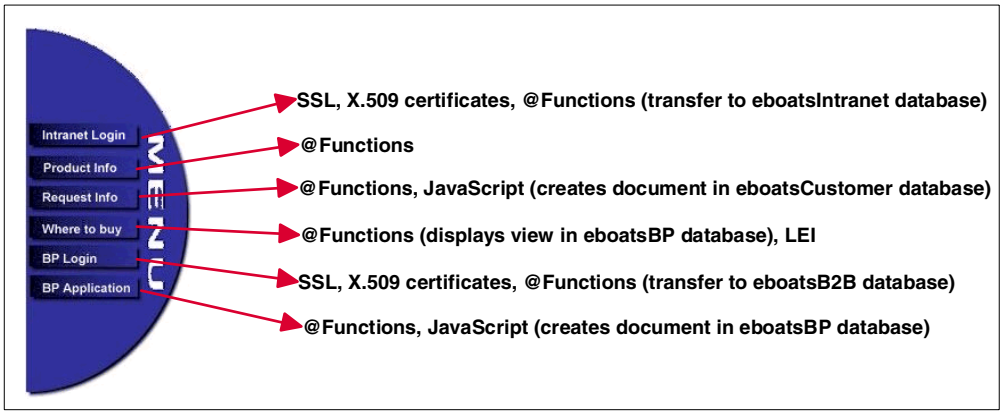


Figure 69. Technologies demonstrated in the Internet database

7.3.2 Business-to-business and extranet database overview

Authorized business partners access this database when they logon to the extranet. From this database, business partners can submit orders through the Internet. Once the business partner submits an order, they can track the status of their order. Business partners can review the current inventory levels for the entire product line. Business partners can view the entire customer list.

This database demonstrates the use of:

- LotusScript
- LS:DO
- @functions
- @DB functions
- LSX LC
- Passthru HTML
- JavaScript
- Java servlet
- Calling stored procedures on the AS/400 system
- CGI variables
- SSL
- X.509 certificates
- DECS

Figure 70 on page 158 demonstrates the technologies used in this database and the relationship with other databases.

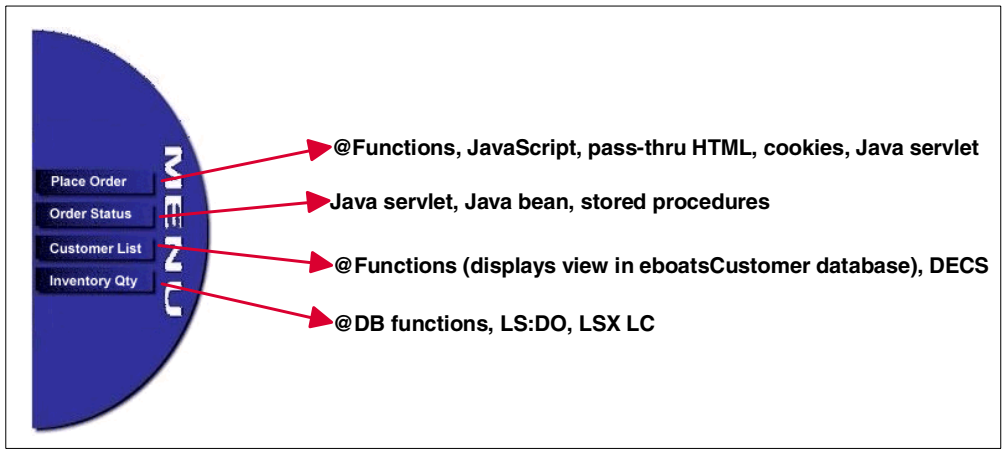


Figure 70. Technologies demonstrated in the business-to-business/extranet database

7.3.3 Intranet database overview

eBoats International employees access this database when they logon to the intranet. Employees may view all orders or view orders for a selected business partner. Order information is displayed using XML, which requires the user to use Microsoft Internet Explorer 5.0 or higher.

Note

Microsoft Internet Explorer 5.0 (or higher) is needed to display information presented with XML. It was the only Web browser that offered integrated support for XML at the time the redbook was written.

If a user displays all orders, all orders that are contained in the Domino database are displayed using XML. When a user selects a business partner number, the data is dynamically displayed based on a query to a DB2 UDB for AS/400 table using XML. In this later example, the data is not contained in the Domino database.

This database demonstrates the use of:

- XML
- XSL style sheets
- Domino workflow
- LotusScript
- @DB functions
- LEI
- Java agent

Figure 71 demonstrates the technologies used in this database and the relationship with other databases.

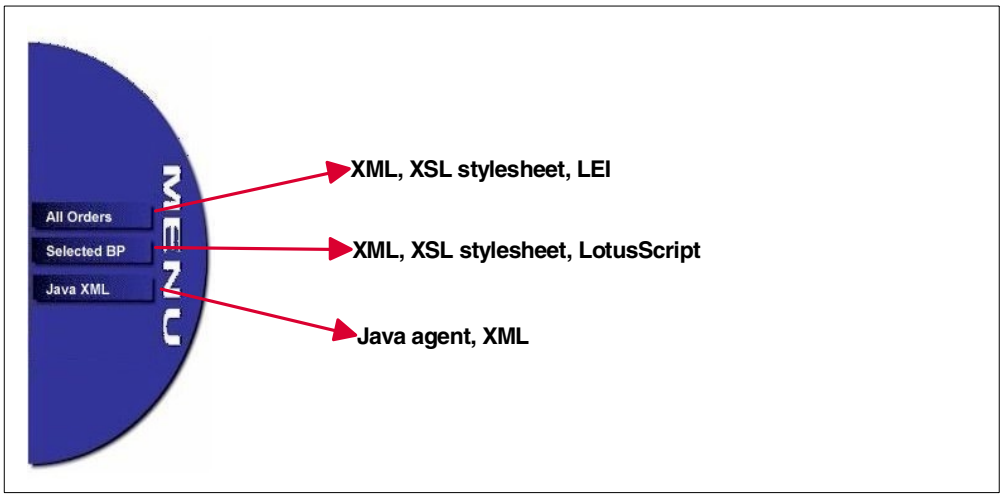


Figure 71. Technologies demonstrated in the intranet database

7.3.4 Business partner database overview

This database provides different functionality when accessed from a browser or from a Notes client. From a browser, a user can only submit a new business partner application.

When a business partner application is submitted from the Internet, Domino workflow capabilities route the application to the appropriate persons within eBoats. Applications are either approved or rejected. The applicant receives notification through e-mail of the final status of the application. The approval process can only be accomplished from a Notes client.

This database demonstrates the use of:

- JavaScript
- LS:DO
- Java servlet
- @functions
- Domino workflow

7.3.5 Customer database overview

All requests for information are stored in this database. Using Domino workflow capabilities, all requests are e-mailed to the nearest business partner who is responsible for contacting the customer directly with answers to their information request.

Business partners are shown a view in this database when they want to access eBoats International's customer list.

This database demonstrates the use of:

- JavaScript
- @functions
- Domino workflow

7.4 Application workflow

Note

For complete sources of all the code examples shown throughout this redbook, please refer to Appendix G, “Using the additional material” on page 403, for instructions on how to access the redbook Web site and download the example Domino databases and DB2 UDB for AS/400 tables.

Domino workflow capabilities are used when a new business partner application is submitted and when a customer requests information. The following sections describe the routing process implemented in the sample databases.

7.4.1 Customer information request workflow

Any user that visits eBoats International Web site can request more information on the company, the products, or eBoats International’s business partners. To request information, users click the Request Info link on the navigator. Once the user fills out the form, they click the Submit button. The form is first validated to make sure the user has entered information into every required field that is marked in red letters. If the forms is successfully validated, a LotusScript agent is executed. This agent creates a new record for the customer in the CUSTOMER DB2 UDB for AS/400 table. Figure 72 shows the workflow for a customer information request.

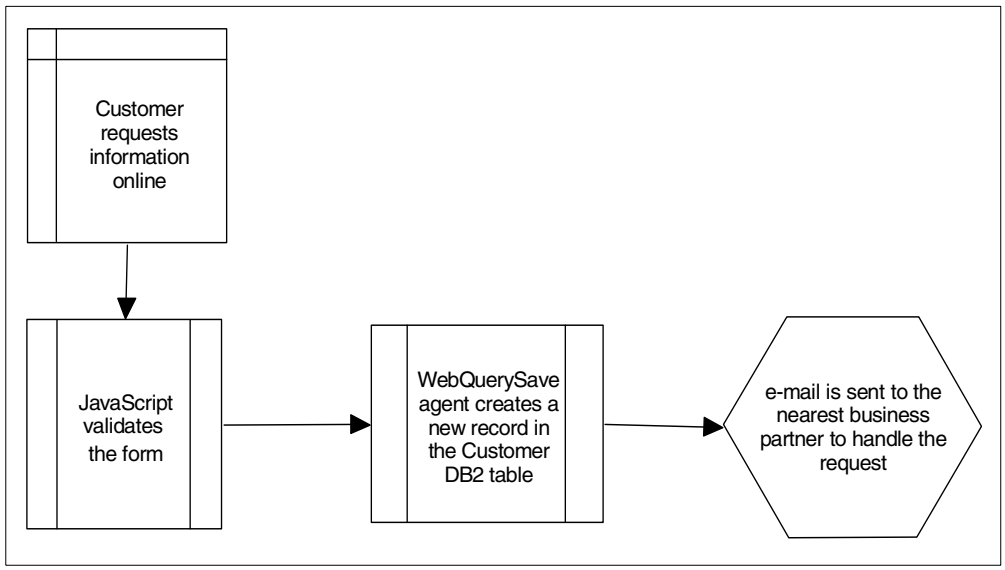


Figure 72. Customer information request workflow

7.4.2 Business partner application workflow

eBoats International only sells its products through its business partner channel. Companies that want to become a business partner can submit an application on the Internet and receive e-mail notification of the approval or rejection of the application within one week.

To start the application process, a user clicks the BP Application link in the navigator. The user is required to fill out all fields that are marked in red. After completing the form, the user clicks the Submit button. This first validates the form to verify that the user has entered information into all required fields. Once the form is validated, a LotusScript LS:DO agent sends an e-mail notification to the appropriate persons within eBoats International. This person is responsible for reviewing the application and then either approving or rejecting the application. If the application is approved, an e-mail notification is sent to the applicant with their new business partner number, user ID and password. A LotusScript agent creates a new record of the business partner in the BPINFO DB2 UDB for AS/400 table and registers the business partner in the secondary Domino Directory. Figure 73 shows the workflow when a user submits a Business Partner application.

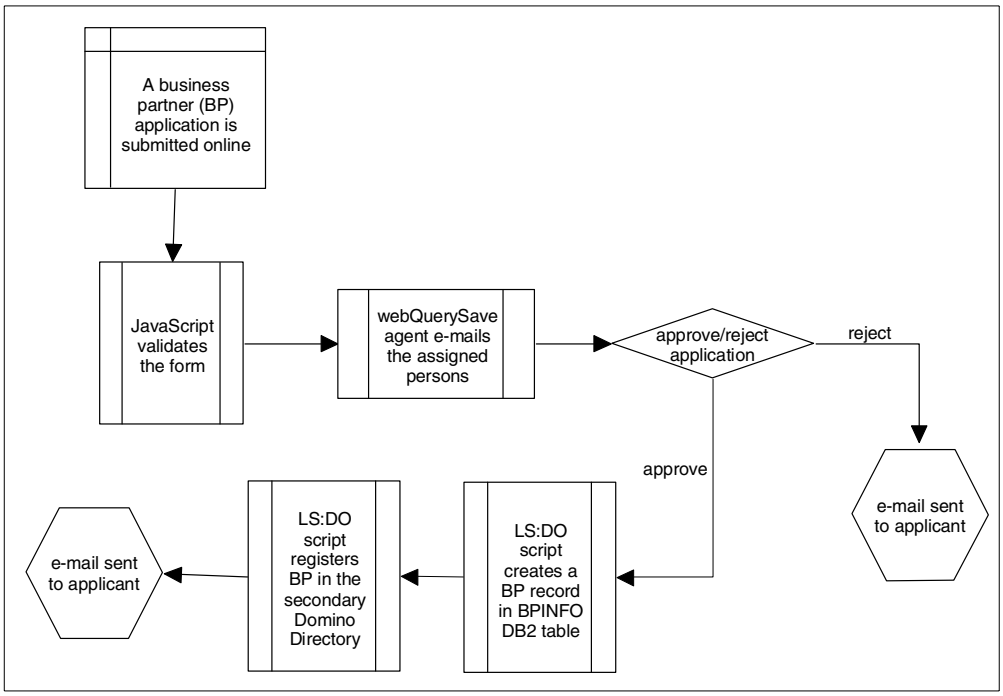


Figure 73. Business partner application workflow

7.5 DB2 UDB for AS/400 tables

This section describes the layout of the four DB2 UDB for AS/400 tables used by the sample application. The four tables are:

- **BPINFO**: Contains information about all registered business partners
- **CUSTOMER**: Contains information about all customers who have requested information
- **ORDERINFO**: Contains all orders placed by the business partners
- **PRODUCTS**: Contains the current inventory level of the products

Their relationships are shown in Figure 74 on page 162.

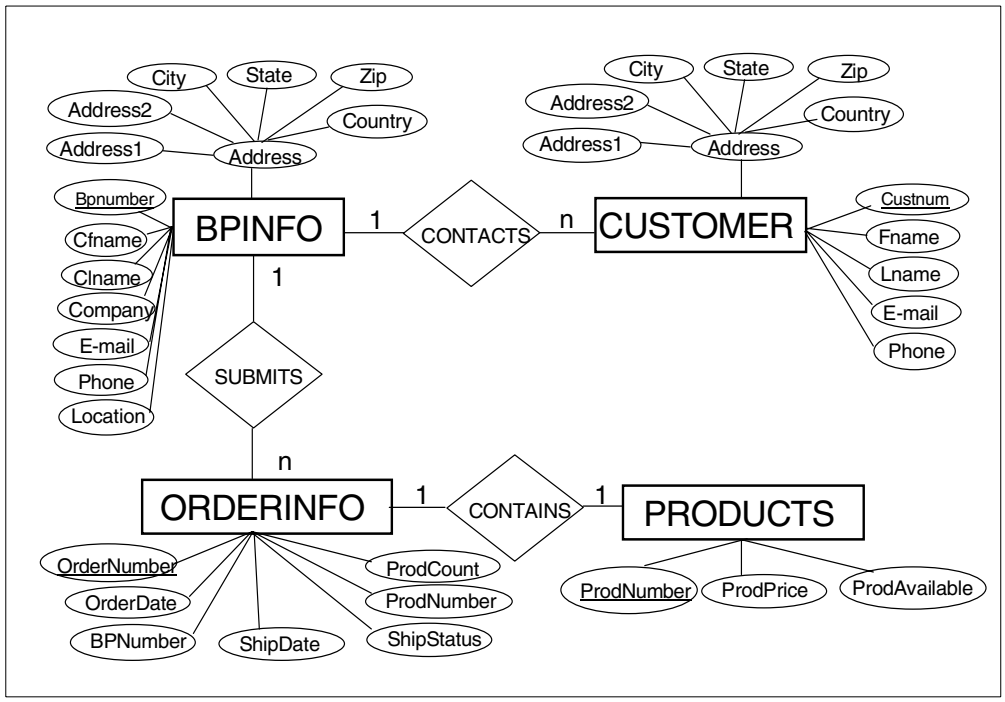


Figure 74. Project's DB2 UDB for AS/400 tables entity relationship diagram

Table 17 through Table 20 describe the format of each of the DB2 UDB for AS/400 tables.

The BPINFO table (Table 17) contains information on all authorized business partners.

Table 17. Business partner table layout (BPINFO)

Field name	Real name	Type	Length
COMPANY	Company Name	Character	40
ADDRESS1	Address Line1	Character	20
ADDRESS2	Address Line 2	Character	20
CITY	CITY	Character	20
STATE	State/Province	Character	20
COUNTRY	Country	Character	20
PHONE	Telephone Number	Character	20
CFNAME	Contact First Name	Character	10
CLNAME	Contact Last Name	Character	10
E-MAIL	e-mail Address	Character	30
ZIP	Postal Code	Character	7
BPNUMBER	Business Partner Number	Character	4
LOCATION	Location	Character	15

The CUSTOMER table (Table 18) contains information on all customers who have requested information.

Table 18. Customer table layout (CUSTOMER)

Field name	Real name	Type	Length
FNAME	First Name	Character	10
LNAME	Last Name	Character	10
ADDRESS1	Address Line 1	Character	20
ADDRESS2	Address Line 2	Character	20
CITY	City	Character	20
STATE	State/Province	Character	20
COUNTRY	Country	Character	20
E-MAIL	e-mail Address	Character	30
PHONE	PHone Number	Character	20
ZIP	Postal Code	Character	7
CUSTNUM	Customer Number	Character	10

The ORDERINFO table (Table 19) contains every order submitted by the business partners.

Table 19. Order table layout (ORDERINFO)

Field name	Real name	Type	Length
ORDERNUMBER	Order Number	Character	6
ORDERDATE	Order Date	Character	8
BPNUMBER	Business Partner Number	Character	4
SHIPDATE	Ship Date	Character	8
PRODNUMBER	Product Number	Character	9
PRODCOUNT	Quantity	Numeric	4
SHIPSTATUS	Order Status	Character	1
PRICE	Price	Numeric	12

The PRODUCTS table (Table 20) contains the current inventory level for all products.

Table 20. Products table layout (PRODUCTS)

Field name	Real name	Type	Length
PRODNUMBER	Product Number	Character	9
PRODPRICE	Product Price	Numeric	12
PRODAVAILABLE	Quantity Available	Numeric	8

7.6 Database terminology

This redbook concentrates on using the AS/400 system as a database server in a client/server environment. In some cases, we use SQL to access the AS/400 database. In other cases, we use native database access.

The terminology used for the database access is different in both cases. In Table 21, you find the native term and the corresponding SQL term.

Table 21. Database terminology

AS/400 native	SQL
Library	Collection
Physical file	Table
Field	Column
Record	Row
Logical file	View or index

Chapter 8. Sample project development

The sample application developed to demonstrate the technologies discussed in this book consist of five Domino databases and four DB2 UDB for AS/400 tables. This chapter describes the application development that was done on the AS/400 system and in the Domino databases. The following sections describe the development process.

8.1 The AS/400 part

eBoats International already maintains its current inventory levels, information on its authorized Business Partners and customers, and all orders in DB2 UDB for AS/400 tables. As part of the development process, we decided to create Domino databases that will interface with these existing legacy systems. The Domino databases will provide workflow and content management while DB2 continues to store mission critical data.

In this section, we describe the format of the DB2 UDB for AS/400 tables, how to create the tables, and how to create new records in the tables. Since the Domino databases will closely integrate with these tables, we demonstrate how to query the tables and present the results in a Domino database.

We also describe the security for those tables.

8.1.1 DB2 UDB for AS/400 tables and stored procedures

The detailed database structure for each table is described in 7.5, “DB2 UDB for AS/400 tables” on page 161. SQL statements are used to create a new table on the AS/400 system. The following sections describe how to create a new table on the AS/400 system and the format for creating a stored procedure.

8.1.1.1 Creating a table

Before you create a table, you should first create a collection on the AS/400 system. A collection is a special AS/400 library where a journal and journal receivers are created automatically by the operating system. Meta data or dictionary files in this library describe the tables you create. You must have a collection if you want to access DB2 UDB for AS/400 from Domino and Java. To create a collection, enter the following commands on the AS/400 system:

1. Logon to your AS/400 system. In our example, we assume that you are using an AS/400 user profile called EBOWNER. This profile will appear as the owner of the collection and of the objects contained in the collection (tables, views, stored procedure, and so on).
2. Enter this command at the AS/400 command prompt:

   ```
   STRSQL
   ```

3. Enter this command at the SQL statements command prompt:

   ```
   CREATE COLLECTION <collection name>
   ```

   The collection name should be less than 10 characters. For the sample application, we created a collection named eBoats.

After you create a Collection, you can create a table using standard SQL statements. The sample application uses four tables whose structure is defined in

7.5, “DB2 UDB for AS/400 tables” on page 161. The following sections show the necessary SQL statements to create all of these tables.

Note

You can select whether the SQL naming convention or the system naming convention is to be used for this session. The system naming convention uses a “/” character while the SQL naming convention uses a “.” character. To change the naming convention, follow these steps:

1. Enter the command:

   `STRSQL`

2. Press the F13 function key.
3. Enter `1` and press the Enter key.
4. Change the naming convention to `*SQL` and press the Enter key.
5. Press F3 to finish.

Order information table

The ORDERINFO table describes all orders. This table has fields containing the order number, order loaded date, Business Partner number, product number, product quantity, predicted shipping date, order unit price, and shipment status.

Here is the SQL statement to create the ORDERINFO table:

```
CREATE TABLE EBOATS.ORDERINFO
 (ORDERNUMBER CHAR (6 ) NOT NULL  WITH DEFAULT,
  ORDERDATE CHAR (8 ) NOT NULL  WITH DEFAULT,
  BPNUMBER  CHAR (4 ) NOT NULL  WITH DEFAULT,
  SHIPDATE CHAR (8 ) NOT NULL  WITH DEFAULT,
  PRODNUMBER CHAR (9 ) NOT NULL  WITH DEFAULT,
  PRODCOUNT NUMERIC (4 ) NOT NULL  WITH DEFAULT,
  SHIPSTATUS CHAR (1 ) NOT NULL  WITH DEFAULT,
  PRICE NUMERIC(12,2) NOT NULL  WITH DEFAULT)
```

Enter information above on the command line of the Interactive SQL Session and press the Enter key. If the table is created successfully, you will see the following message:

```
Table ORDERINFO created in EBOATS
```

Business partner information table

The BPINFO table describes our business partners. This table has fields containing company name, address, phone number, e-mail address, and contact name.

Here is the SQL statement to create the BPINFO table:

```
CREATE TABLE EBOATS.BPINFO
(COMPANY CHAR (40 ) NOT NULL  WITH DEFAULT,
ADDRESS1 CHAR (20 ) NOT NULL  WITH DEFAULT,
ADDRESS2 CHAR (20 ) NOT NULL  WITH DEFAULT,
CITY CHAR (20 ) NOT NULL  WITH DEFAULT,
STATE CHAR (20 ) NOT NULL  WITH DEFAULT,
COUNTRY CHAR (20 ) NOT NULL  WITH DEFAULT,
PHONE CHAR (20 ) NOT NULL  WITH DEFAULT,
CFNAME CHAR (10 ) NOT NULL  WITH DEFAULT,
CLNAME CHAR (10 ) NOT NULL  WITH DEFAULT,
EMAIL CHAR (30 ) NOT NULL  WITH DEFAULT,
ZIP CHAR (7 ) NOT NULL  WITH DEFAULT,
BPNUMBER CHAR (4 ) NOT NULL  WITH DEFAULT,
LOCATION CHAR(15 ) NOT NULL  WITH DEFAULT)
```

Customer Information table

The CUSTOMER table describes our customers. This table has fields containing customer name, address, phone number, e-mail address, and customer unique number.

Here is the SQL statement to create the CUSTOMER table:

```
CREATE TABLE EBOATS.CUSTOMER
(FNAME CHAR (10 ) NOT NULL  WITH DEFAULT,
LNAME CHAR (10 ) NOT NULL  WITH DEFAULT,
ADDRESS1 CHAR (20 ) NOT NULL  WITH DEFAULT,
ADDRESS2 CHAR (20 ) NOT NULL  WITH DEFAULT,
CITY CHAR (20 ) NOT NULL  WITH DEFAULT,
STATE CHAR (20 ) NOT NULL  WITH DEFAULT,
COUNTRY CHAR (20 ) NOT NULL  WITH DEFAULT,
EMAIL CHAR (30 ) NOT NULL  WITH DEFAULT,
PHONE CHAR (20 ) NOT NULL  WITH DEFAULT,
ZIP CHAR (7 ) NOT NULL  WITH DEFAULT,
CUSTNUM CHAR(10 ) NOT NULL  WITH DEFAULT)
```

Product inventory table

The PRODUCTS table describes current inventory levels for our products. This table has fields containing product number, product price, and current inventory level.

Here is the SQL statement to create the PRODUCTS table:

```
CREATE TABLE EBOATS.PRODUCTS
(PRODNUMBER CHAR (9 ) NOT NULL  WITH DEFAULT,
PRODPRICE NUMERIC(12,2 ) NOT NULL  WITH DEFAULT,
PRODAVAILABLE NUMERIC(8,0) NOT NULL  WITH DEFAULT)
```

8.1.1.2 Creating a stored procedure

There are many methods that can be used to access information contained in the DB2 UDB for AS/400 tables. In some situations, it is more efficient to access the tables using a stored procedure. Stored procedures can be written in C, Cobol, RPG, or SQL scripts. You can pass parameters into the stored procedure, which will return the results to the calling program. The results are returned from the OUT parameters or the return values in the stored procedure. You can also get a result set from the stored procedure.

In the sample application, we create a stored procedure from an SQL script. This stored procedure will query the DB2 UDB for AS/400 table and return a result set.

The SQL statements to create this stored procedure are shown here:

```
create procedure eboats.queryOrder( in inum char(4))
result set 1
language sql
begin
    DECLARE c1 CURSOR FOR
        SELECT ORDERNUMBER, ORDERDATE, BPNUMBER, SHIPDATE,
               PRODNUMBER, PRODCOUNT, SHIPSTATUS, PRICE
        FROM eboats.orderinfo WHERE BPNUMBER = inum;
    open c1;
    set result sets cursor c1;
end
```

For more detailed information on creating a stored procedure on the AS/400, please refer to the manual *DB2 UDB for AS/400 SQL Programming*, SC41-5611.

8.1.2 Security for the tables and stored procedure

A user profile is created on the AS/400 system to be used in all database integration functions demonstrated in the sample application. In the sample code

published in this redbook, as well as in DECS or LEI connection documents, the user profile is simply USERID, and the password is PASSWORD. Note that in the real world, nobody would use such a user profile and password.

This user profile can be created using the Create User Profile (CRTUSRPRF) command from a 5250 display, for example:

```
CRTUSRPRF USRPRF(USERID) PASSWORD(PASSWORD) USRCLS(*USER) TEXT('eBoats
application user)
```

Or, the user profile can be created by using AS/400 Operations Navigator.

By default, the eBoats collection is created with a public access set to *CHANGE. You need to set it to *USE, which is sufficient to give access to the objects inside the collection. To edit the permissions to the eBoats library from a 5250 display, you can use the Work with Objects (WRKOBJ) command:

```
WRKOBJ OBJ(QSYS/EBOATS)
```

Enter 2 (Edit authority) in front of the EBOATS. Type *LIB (*LIB stands for library, which is the “container” for the SQL collection). Set the public access to *USE, as shown in Figure 75.

```
                          Edit Object Authority

Object . . . . . :   EBOATS          Owner . . . . . :   EBOWNER
  Library  . . . . :     QSYS        Primary group . . . :   *NONE
Object type  . . . . :   *LIB

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . .   *NONE

                        Object
User        Group       Authority
EBOWNER                 *ALL
*PUBLIC                 *USE

                                                                      Bottom
F3=Exit   F5=Refresh   F6=Add new users   F10=Grant with reference object
F11=Display detail object authorities     F12=Cancel   F17=Top   F18=Bottom
(C) COPYRIGHT IBM CORP. 1980, 1999.
```

Figure 75. Edit Object Authority for the EBOATS collection from a 5250 display

You can now consider securing the eBoats collection tables. Again, this can be done from a 5250 display or from AS/400 Operations Navigator.

From a 5250 display, you can list all the objects contained in the collection EBOATS using the Work with Objects (WRKOBJ) command, for example by entering:

```
WRKOBJ OBJ(EBOATS/*ALL)
```

```
                              Work with Objects

Type options, press Enter.
  2=Edit authority        3=Copy   4=Delete   5=Display authority   7=Rename
  8=Display description   13=Change description

Opt  Object      Type      Library     Attribute   Text
     QUERYORDER  *PGM      EBOATS      CLE         SQL PROCEDURE QUERYORDER
     QSQJRN0001  *JRNRCV   EBOATS                  COLLECTION - created by SQL
     QSQJRN0002  *JRNRCV   EBOATS                  COLLECTION - created by SQL
     QSQJRN0003  *JRNRCV   EBOATS                  COLLECTION - created by SQL
     QSQJRN0004  *JRNRCV   EBOATS                  COLLECTION - created by SQL
     QSQJRN0005  *JRNRCV   EBOATS                  COLLECTION - created by SQL
     QSQJRN0006  *JRNRCV   EBOATS                  COLLECTION - created by SQL
     QSQJRN      *JRN      EBOATS                  COLLECTION - created by SQL
     BPINFO      *FILE     EBOATS      PF
     CUSTOMER    *FILE     EBOATS      PF
     ORDERINFO   *FILE     EBOATS      PF
                                                                        More...
Parameters for options 5, 7 and 13 or command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F11=Display names and types
F12=Cancel   F16=Repeat position to   F17=Position to
```

Figure 76. EBOATS objects listed using the Work with Objects command

You can use option 2 (Edit authority) to edit the object authority for each object that needs it. Figure 77 shows the changed authorities for the table, BPINFO, owned by the AS/400 user profile EBOWNER. *CHANGE access is given to AS/400 user USERID. Conversely, the new authority for *PUBLIC is set *EXCLUDE, to prevent any other AS/400 user from reading or updating it, instead of *CHANGE.

```
                           Edit Object Authority

Object . . . . . :   BPINFO          Owner  . . . . . :   EBOWNER
  Library  . . . . :     EBOATS      Primary group  . . :   *NONE
Object type  . . . . :   *FILE

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . .   *NONE

                        Object
User        Group       Authority
EBOWNER                 *ALL
USERID                  *CHANGE
*PUBLIC                 *EXCLUDE




                                                                      Bottom
F3=Exit   F5=Refresh   F6=Add new users   F10=Grant with reference object
F11=Display detail object authorities     F12=Cancel   F17=Top   F18=Bottom
Object authorities changed.
```

Figure 77. Authorities for the BPINFO table changed from a 5250 display

From AS/400 Operations Navigator, click the AS/400 system name. Then, click **Database**, and right-click **Libraries**. Click **Select Libraries to Display**, and enter the library name EBOATS.

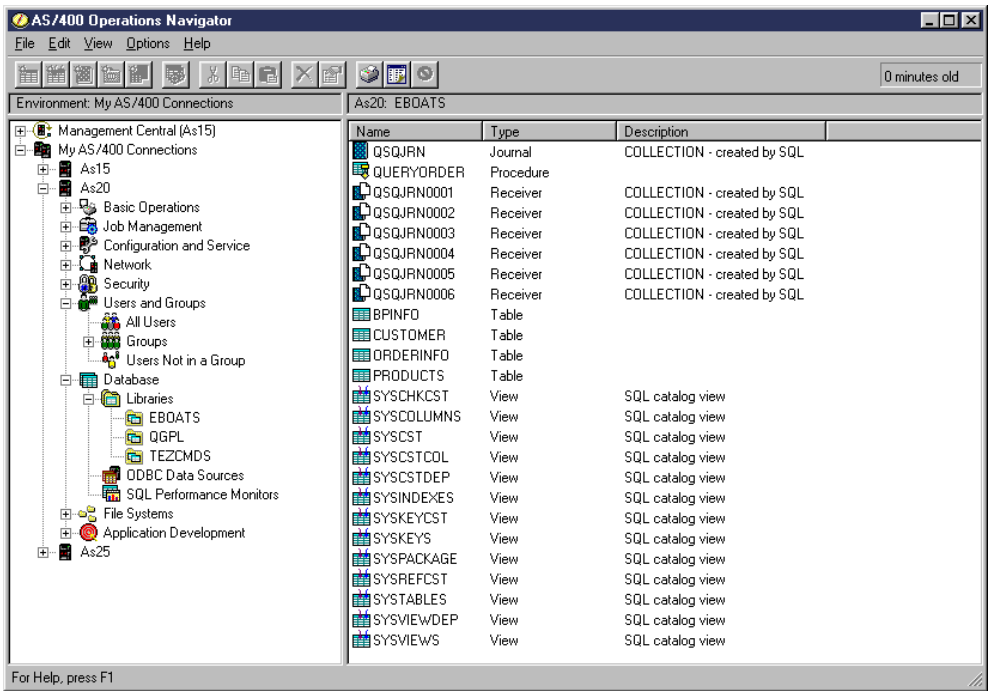


Figure 78. EBOATS objects displayed using AS/400 Operations Navigator

You can right-click **BPINFO**, and then select **Permissions**. Figure 79 shows the next display, which allows you to change permissions again.

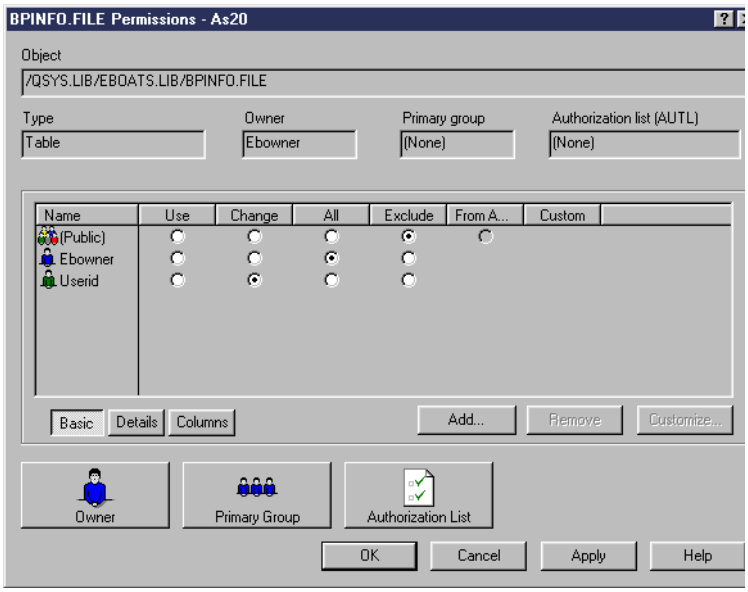


Figure 79. BPINFO table permissions displayed using AS/400 Operations Navigator

For all EBOATS tables, you give *CHANGE permission to user profile USERID. This allows this user to create, update, or delete records in these tables.

The same steps need to be followed to give permissions to the stored procedure QUERYORDER. You give a *USE permission to user profile USERID.

Information about AS/400 security basics can be found in *Security - Basic V4R1*, SC41-5301.

8.2 Domino part

There are five databases for the sample application. Three databases represent the following functional areas of eBoats International:

- Internet (eboatsInternet.nsf)
- Business-to-Business/extranet (eboatsB2B.nsf)
- Intranet (eboatsIntranet.nsf)

The remaining two databases are helper databases that are used to capture customer requests for information (eboatsCustomer.nsf) and business partner applications (eboatsBP.nsf). This section describes the forms, views and programming languages used within each of the five databases.

Note

We assume here that all the sample project's Domino databases are managed by an administrator or any other user defined in the group Administrators that is set as described in 9.1.2, "First administration steps" on page 224.

8.2.1 Internet database: eboatsInternet.nsf

This database is eBoats International's Web site. This database provides information about the company's product line, a request for information form, the location of business partners worldwide, an online application to become a business partner, the login point for business partners, and the login point for eBoats International employees.

The following sections describe the database's design elements.

8.2.1.1 Security

Everyone accessing this database through the Internet will access it as Anonymous (public user). Anonymous is given Reader access in the database's Access Control List (ACL). Anonymous users are not allowed to create or delete documents. In the Advanced tab of the ACL, maximum Internet name and password access is set to Author. The appropriate administration server is selected for this database. Default Access is set to Reader. The appropriate access levels are given to eBoats International employees.

Every form in this database has a create access level for authors or above. This prohibits Anonymous users from creating any new forms in the database.

8.2.1.2 Database launch properties

If the database is opened in a Web client, the frameset WebFS is displayed. If the database is opened in a Notes client, the frameset NotesFS is displayed. Both framesets are described in 8.2.1.6, "Framesets" on page 183.

Corresponding settings are shown in the database launch properties, as shown in Figure 80. The tab for the database launch properties is the fifth from the left.

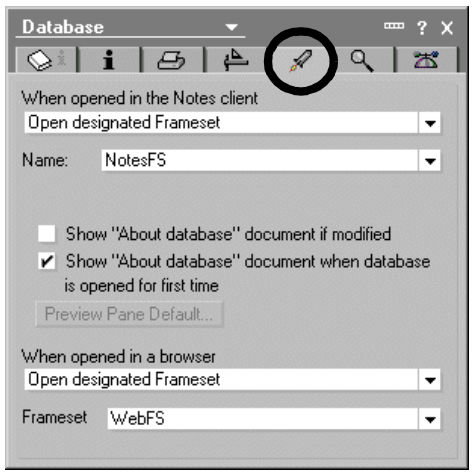


Figure 80. eboatsInternet.nsf database launch properties

Note

To define the database launch properties, select or open the database, and choose **File->Database->Properties**. Click the **Launch** tab.

Choose what to display for a Notes client and what to display for a Web client.

Optionally, choose whether to display the About This Database document when the database opens for the first time or when the About This Database document is modified.

8.2.1.3 Forms

This database uses the following three forms:

- BPLocations
- Product
- $$ViewTemplateDefault

BPLocations form

The BPLocations form contains information about eBoats International business partners. When a business partner application is approved, an agent creates a new record in the BPINFO DB2 UDB for AS/400 table. Information on the business partner is updated from this table through two separate LEI activities. Figure 81 contains the details of the BPLocations form.

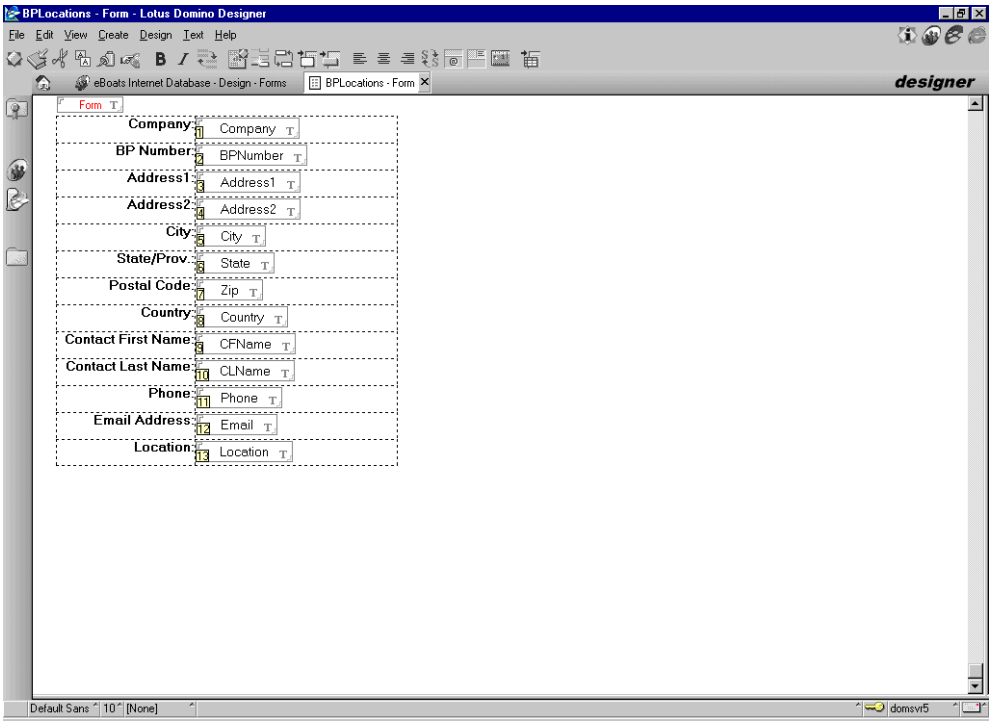


Figure 81. BPLocations form

All fields on the form are text. There is one hidden field called “Form” that is computed when it is composed and contains the value “BPLocations”. The form’s sole purpose is to display information about business partners. The form does not contain any action buttons.

On an hourly basis, LEI does a direct transfer from the BPINFO DB2 UDB for AS/400 table to this database. This activity only transfers the contents of the BPNumber, Company, Country, and Location fields. Only these fields are stored in the Domino database because they are used in a view. The remaining fields are retrieved with an LEI real time activity when a user opens one of the business partner forms. Figure 82 shows the LEI direct transfer activity.

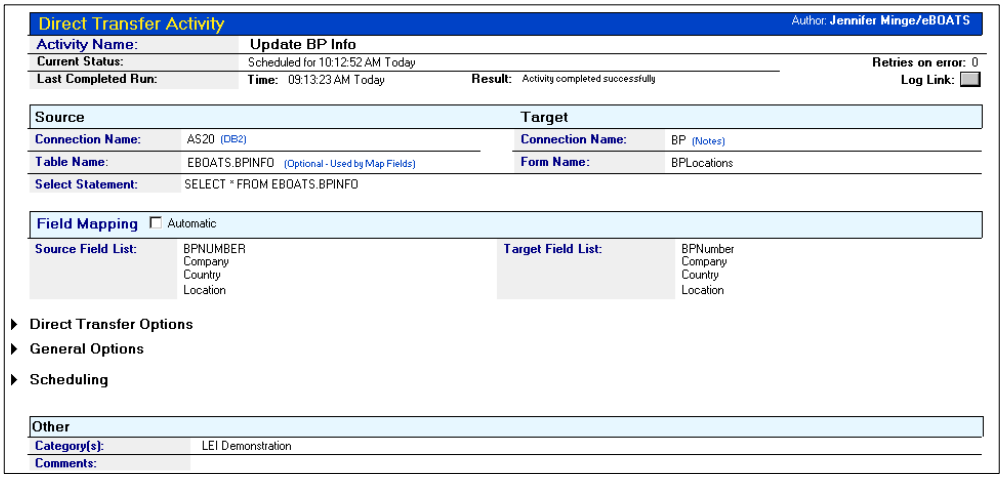


Figure 82. LEI Direct Transfer Activity

This direct transfer activity uses two connection documents called AS20 and BP. Figure 83 shows the details of the AS20 connection document.

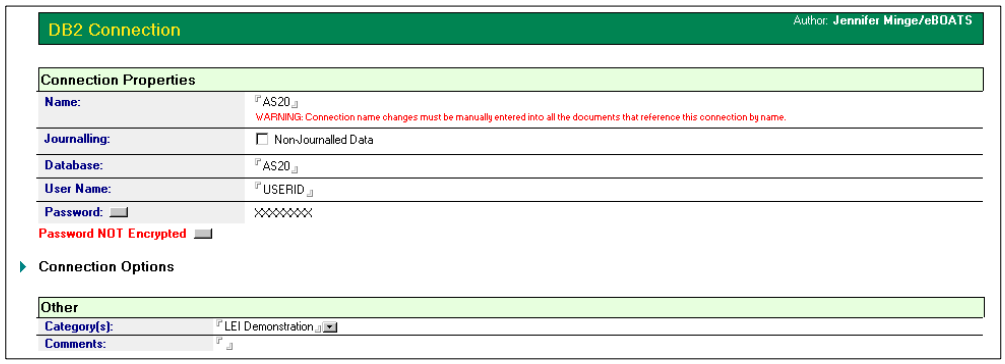


Figure 83. AS20 LEI Connection document

Figure 84 displays the details of the BP connection document.

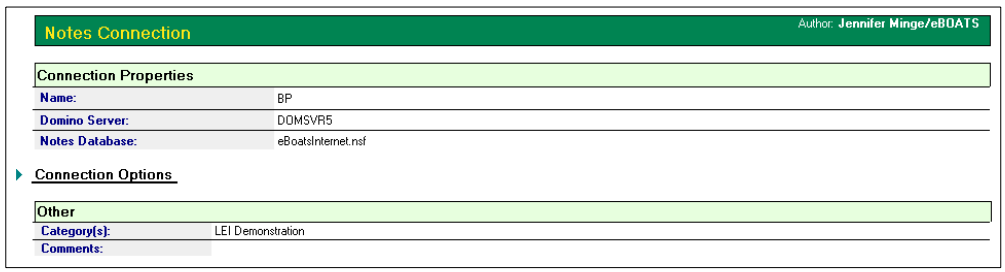


Figure 84. BP LEI Connection document

All other fields are displayed using a LEI RealTime Notes activity when the user opens the form. Figure 85 shows the LEI RealTime Notes activity.

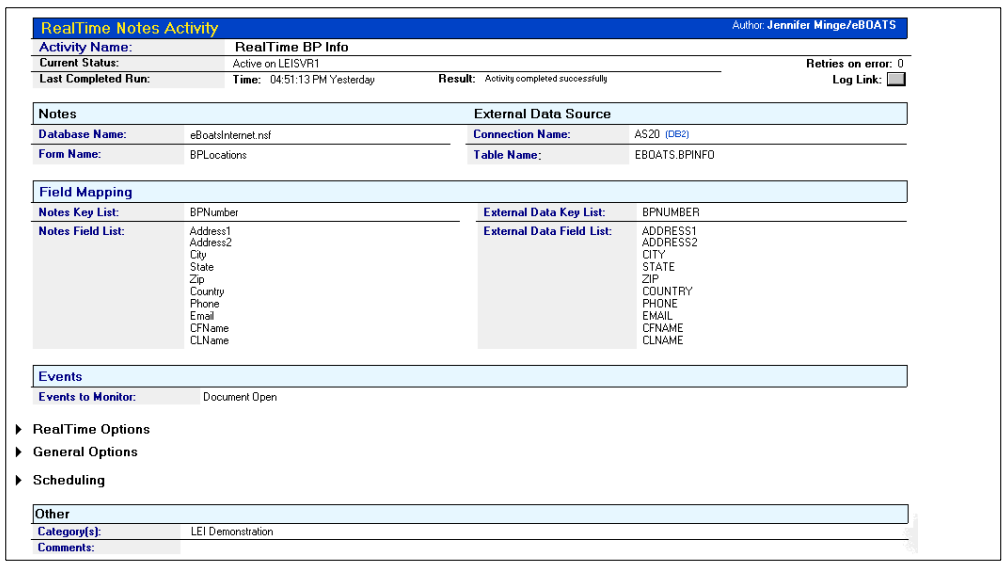


Figure 85. LEI RealTime BP Info activity

Product form

The Product form contains information about the eBoats International's product line. Figure 86 contains the details of the Product form.

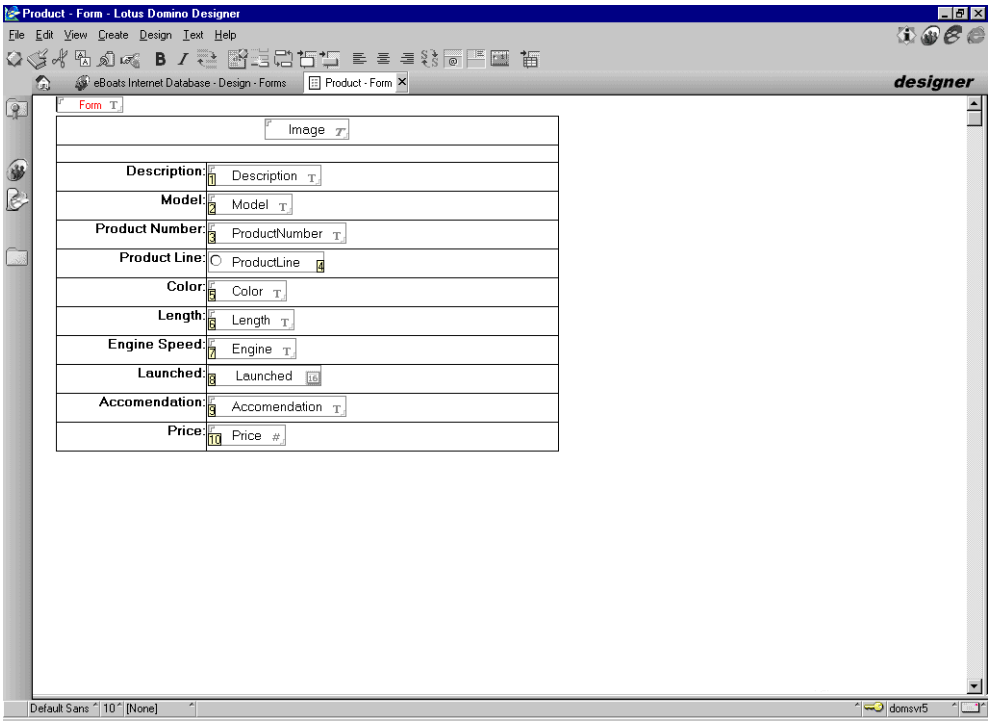


Figure 86. Product form

The Image field contains a picture of the product. The ProductLine field is a radio button that contains the choices Motorboat and Sailboat. The Launched field is a Date Picker field. The Price field is a numeric field. The form field is a hidden field that is computed when composed and contains the value "Product". All other fields are text.

$$ViewTemplateDefault form

The $$ViewTemplateDefault form describes how the products will be displayed to the user. The form contains an embedded view that displays the Products view. The embedded view is set to display as HTML when accessed from the Web. Figure 87 on page 176 contains the details of the $$ViewTemplateDefault form.

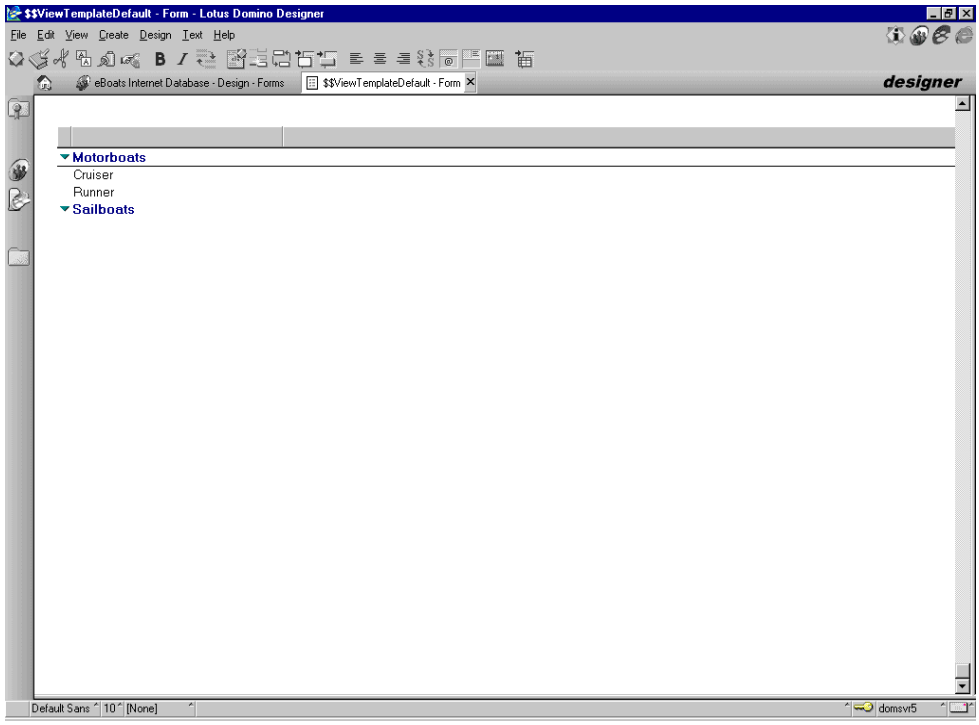


Figure 87. $$ViewTemplateDefault form

8.2.1.4 Pages

The database contains the following four pages:

- AboutUs
- WelcomeBanner
- Outline
- Authors

AboutUs page

This page, displayed when the database first opens, contains information about the database and the technologies that are demonstrated in the project. This page is displayed in the WebFS frameset when the database is launched. Figure 88 shows the details of this page.

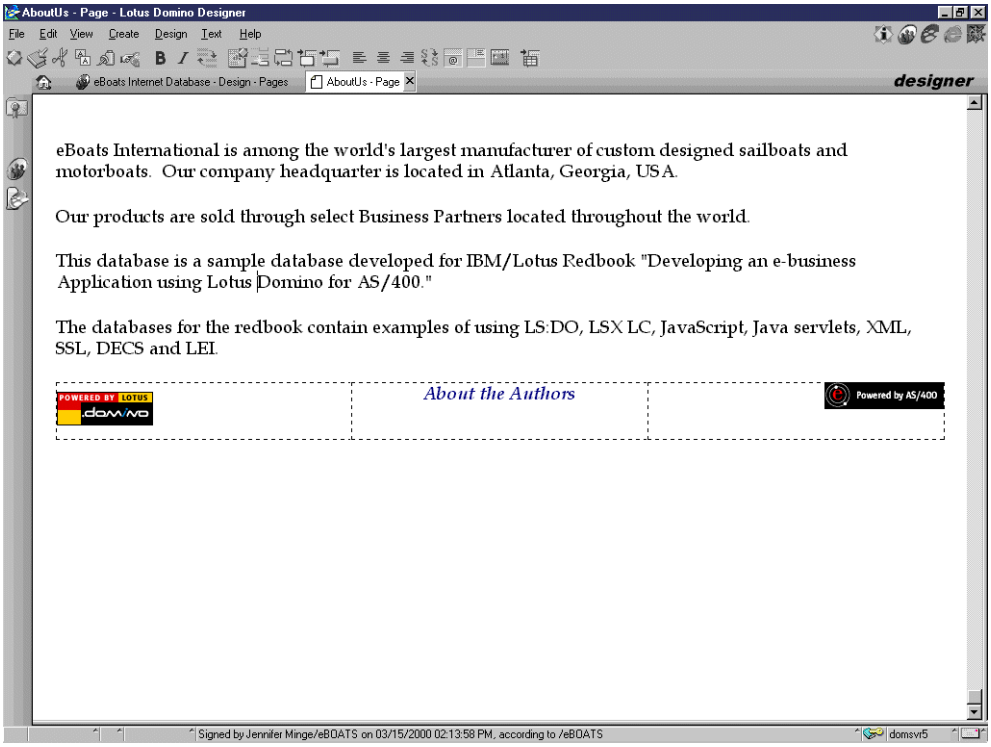


Figure 88. AboutUs page

The onLoad event for this page calls the JavaScript function scroll() which displays a scrolling message across the status line of the browser. This function is contained in the JSHeader. The code for this function is listed in B.1, “AboutUs page JSHeader code” on page 337.

The Unload event of this page clears the status bar of the browser by executing this code:

```
window.status=''
```

WelcomeBanner page

This is the page displayed in the top of the WebFS frameset. This page contains the company name. Figure 89 on page 178 shows the details of this page.

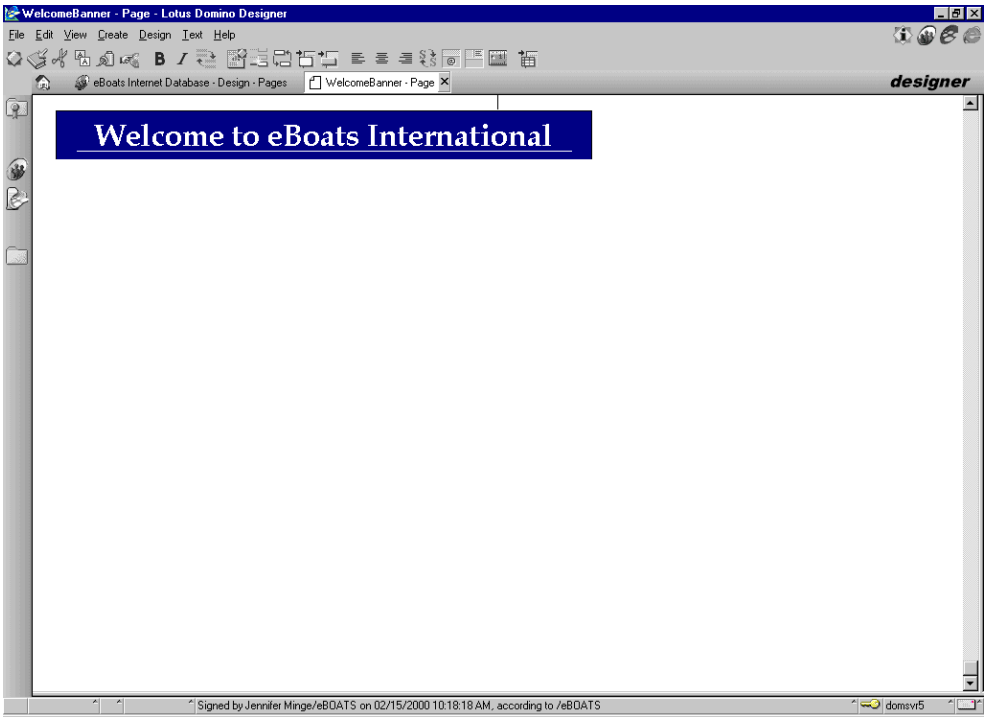


Figure 89. WelcomeBanner page

Outline page

This page, displayed in the frameset NotesFS (described in “NotesFS frameset” on page 183), contains an embedded outline that is used as the navigator for Notes users. Figure 90 shows the details of this page.

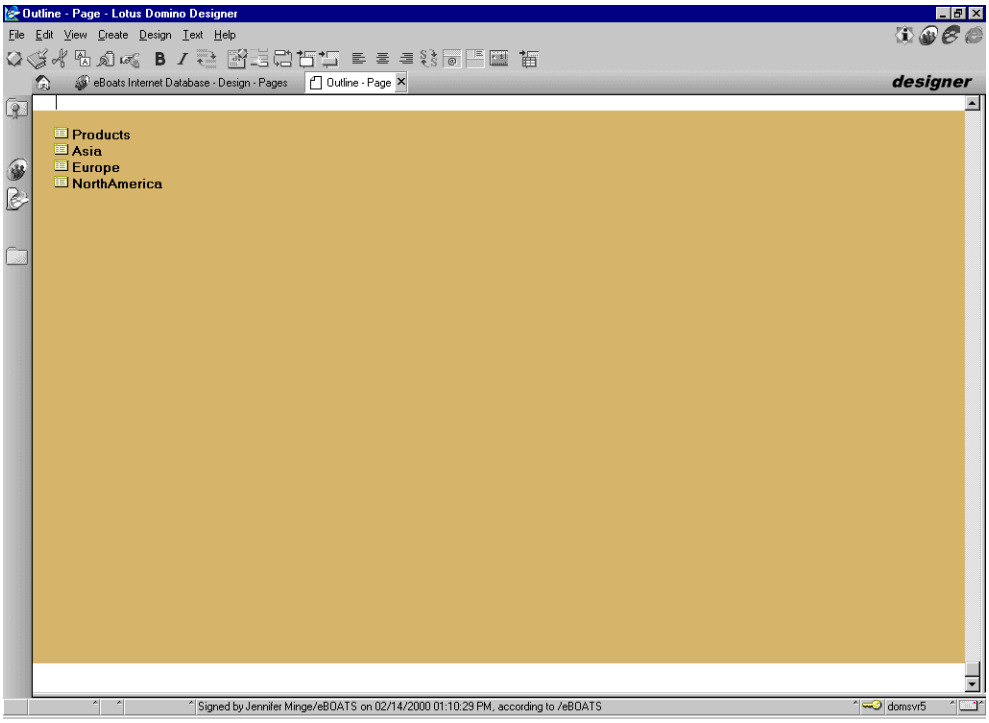


Figure 90. Outline page

Authors page

This page contains five sections containing a picture and biography of the five members who wrote this book and developed the sample applications. Figure 91 shows the details of this page.

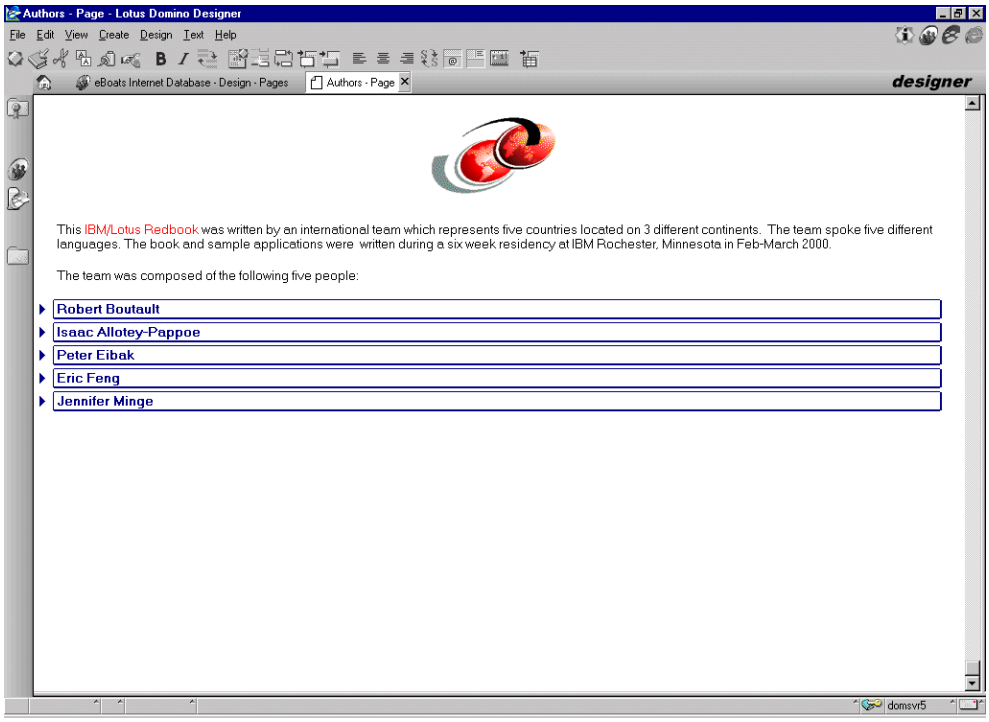


Figure 91. Authors page

8.2.1.5 Navigators

The database contains the following two navigators:

- SideNav
- Map

SideNav navigator

This is the navigator shown in the left frame of the WebFS frameset. This contains links to all of the functions you can accomplish in this database when it is accessed from the Web. Figure 92 on page 180 shows the details of this navigator.

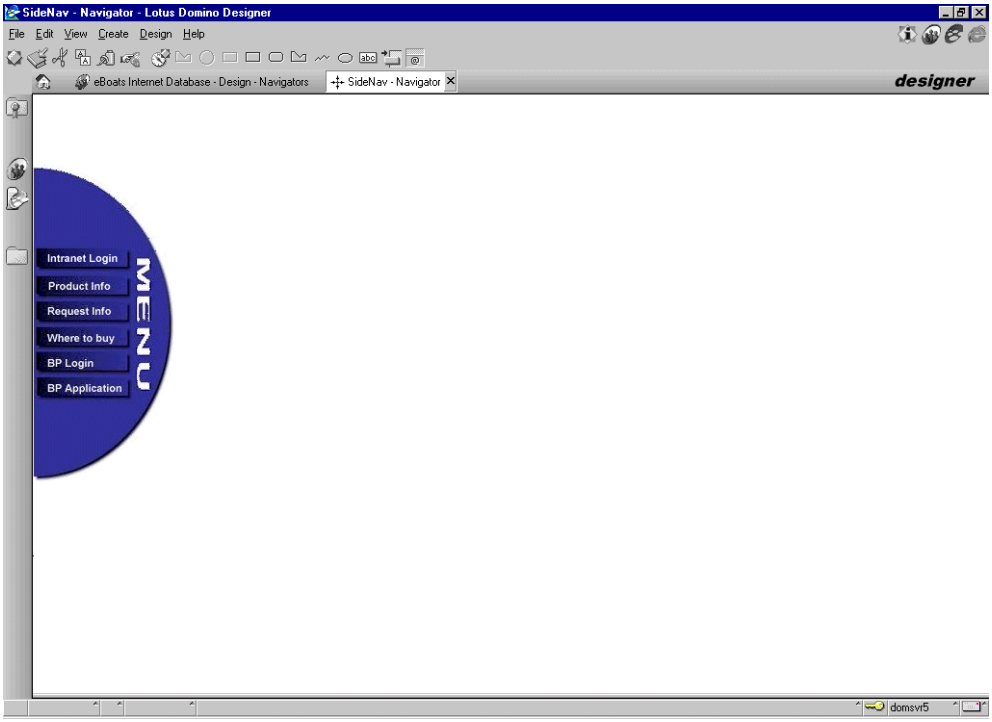


Figure 92. SideNav navigator

From this navigator, visitors can:

- Login to the intranet (option reserved for eBoats International employees)
- Access product information
- Request information
- Access eBoats International distributors information
- Login to the business partners' part of the application (option reserved for eBoats International business partners)
- Apply to become a business partner

Each of these options is described in the following paragraphs.

When a user clicks *Intranet Login*, the browser loads the URL corresponding to the Intranet database eboatsIntranet.nsf. The code is:

```
@SetTargetFrame("_top");
@URLOpen( "/eboatsIntranet.nsf?Open" )
```

When a user clicks *Product Info*, the browser loads the Products view in the current Domino database. The code is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/Products?OpenView";
@URLOpen(url)
```

When a user clicks *Request Info*, the browser loads the Customer form from the Domino database eboatsCustomer.nsf. The code is:

```
@Command([Compose]; "" : "eboatsCustomer.nsf" ; "Customer")
```

When a user clicks *Where to buy*, the browser loads the navigator Map in the current Domino database. The code is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/Map?OpenNavigator";
@URLOpen(url)
```

When a user clicks *BP Login*, the browser loads the URL corresponding to the extranet database eboatsB2B.nsf. The code is:

```
@SetTargetFrame("_top");
@URLOpen( "/eboatsb2b.nsf?Open" )
```

When a user clicks *BP Application*, the browser loads the form BP from the Domino database eboatsBP.nsf. The code is:

```
@SetTargetFrame("_top");
@URLOpen( "/eboatsBP.nsf/BP?OpenForm" )
```

Map navigator

This navigator displays a map of the world. From here, a user can click an area of the world to be shown to the business partners in that area. Figure 93 shows the navigator.

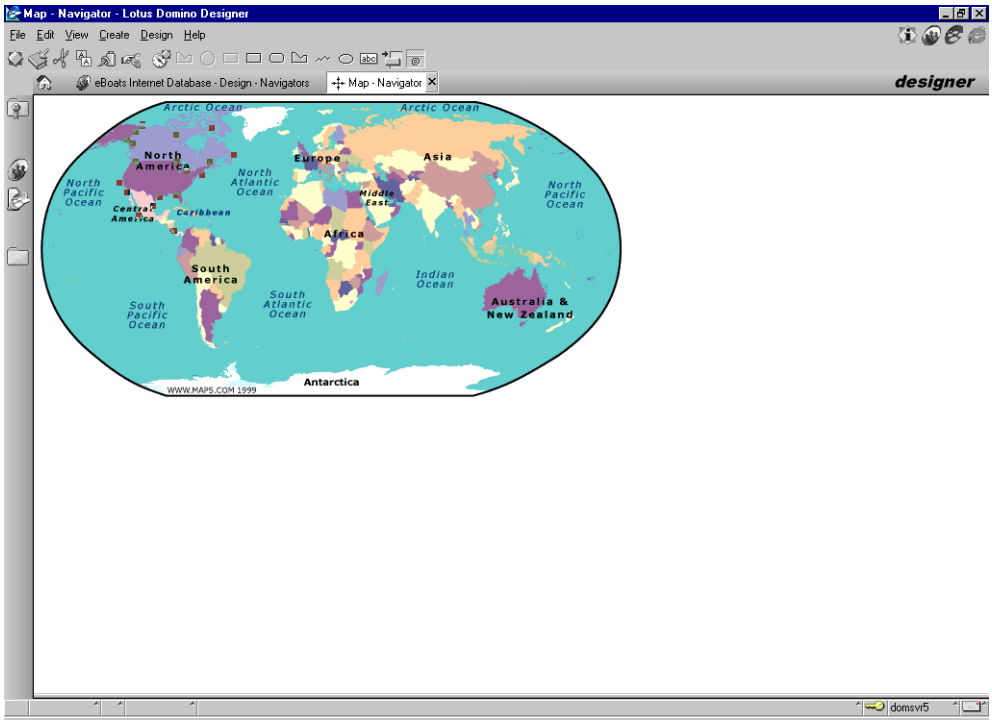


Figure 93. Map navigator

Hotspots are created to represent the different geographical areas where eBoats International is present through its business partners:

- Asia
- Europe
- North America

To create each hotspot, you use the hotspot tool from the SmartIcons palette. The polygon tool allows you to create a line segment each time you click. Double-click

to complete the drawing and close the polygon hotspot. Double-click the hotspot to edit its properties.

For example, the hotspot used for Asia is shown in Figure 94. It has been drawn around the Asian countries where there are eBoats International business partners.

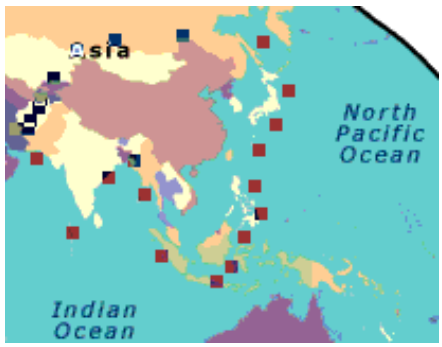


Figure 94. Navigator hotspot

The formula that is used when clicking the hotspot is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/Asia?OpenView";
@URLOpen(url)
```

It displays the view Asia, which can be found in the current Domino database. The view Asia and the two other views are described in 8.2.1.7, “Views” on page 184.

Because the navigator is expected to be used from Web browsers, the “Web browser compatible” navigator property is selected, as shown in Figure 95. With this setting on, Domino can convert the navigator to an HTML image map.

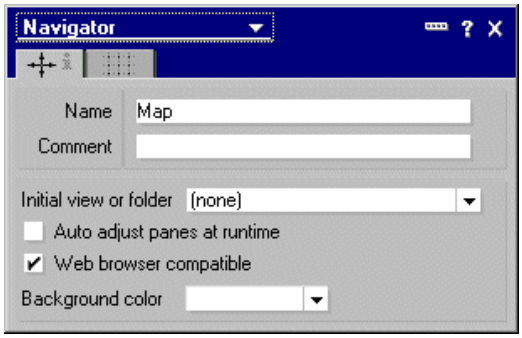


Figure 95. Navigator properties

Note

Navigators on the Web always display as full-screen image maps. To control the size and display of a navigator on the Web, you can embed a navigator in a form.

8.2.1.6 Framesets

The database contains the following two framesets:

- WebFS
- NotesFS

WebFS frameset

This frameset is displayed when the database is opened from a Web browser, as described in 8.2.1.2, “Database launch properties” on page 171. No specific frameset property is defined.

It contains three frames. The left frame displays the SideNav navigator, the top frame displays the WelcomeBanner page, and the right frame displays what is selected from the SideNav navigator. By default, the right frame displays the AboutUs page. Figure 96 shows the details of this frameset.

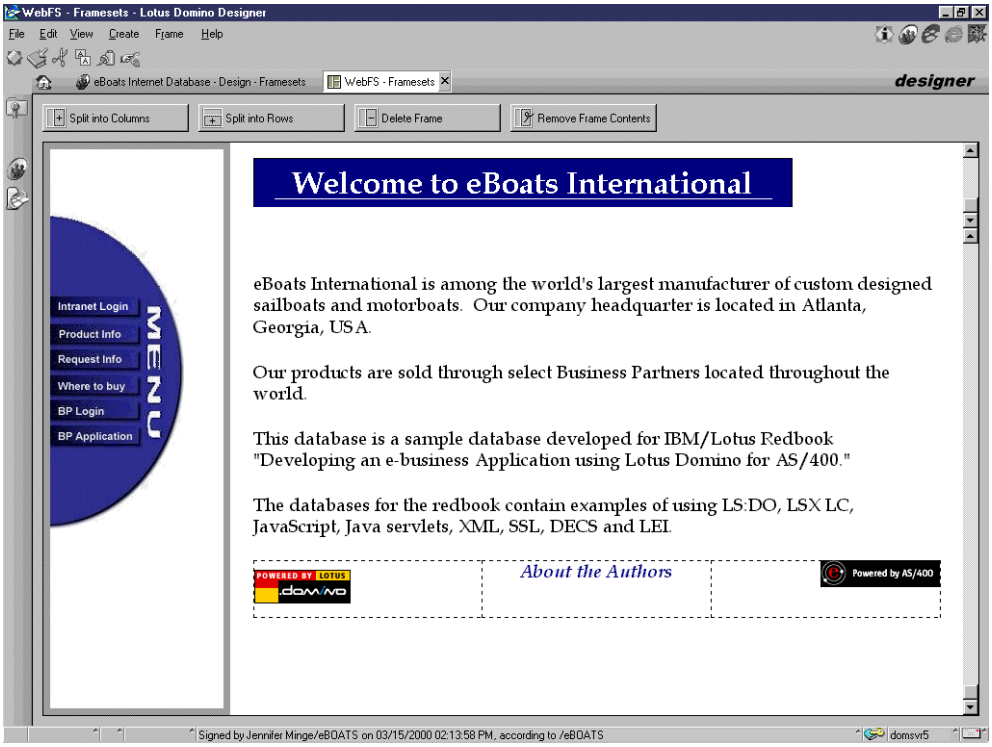


Figure 96. WebFS frameset

NotesFS frameset

This frameset is displayed when the database is opened from a Notes client, as described in 8.2.1.2, “Database launch properties” on page 171.

This frameset has an outline in the left frame and displays views in the right frame. Figure 97 on page 184 shows the details of this frameset. No specific frameset property is defined.

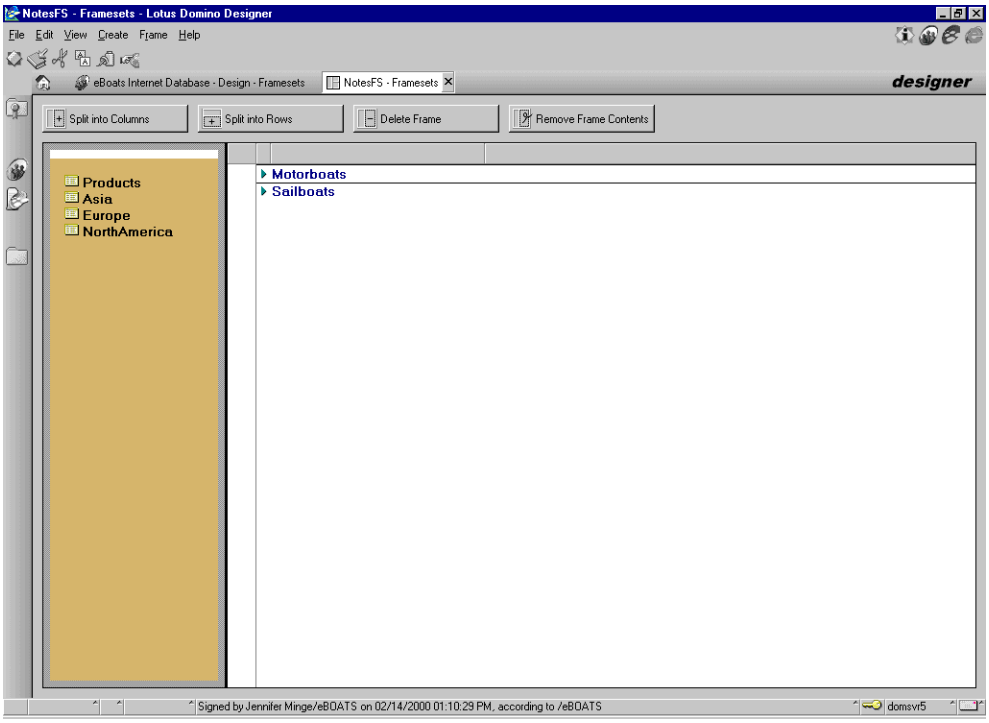


Figure 97. NotesFS frameset

8.2.1.7 Views

The database contains the following five views:

- Asia
- Europe
- North America
- Products
- AllProducts

The Asia view

The Asia view displays a list of all business partners in the Asia region. The view selection formula is:

```
SELECT Form="BPLocations" & Location = "Asia"
```

The view contains two columns. The first column is categorized in ascending order and shows the Country field. The second column displays the Company field. Figure 98 shows this view.

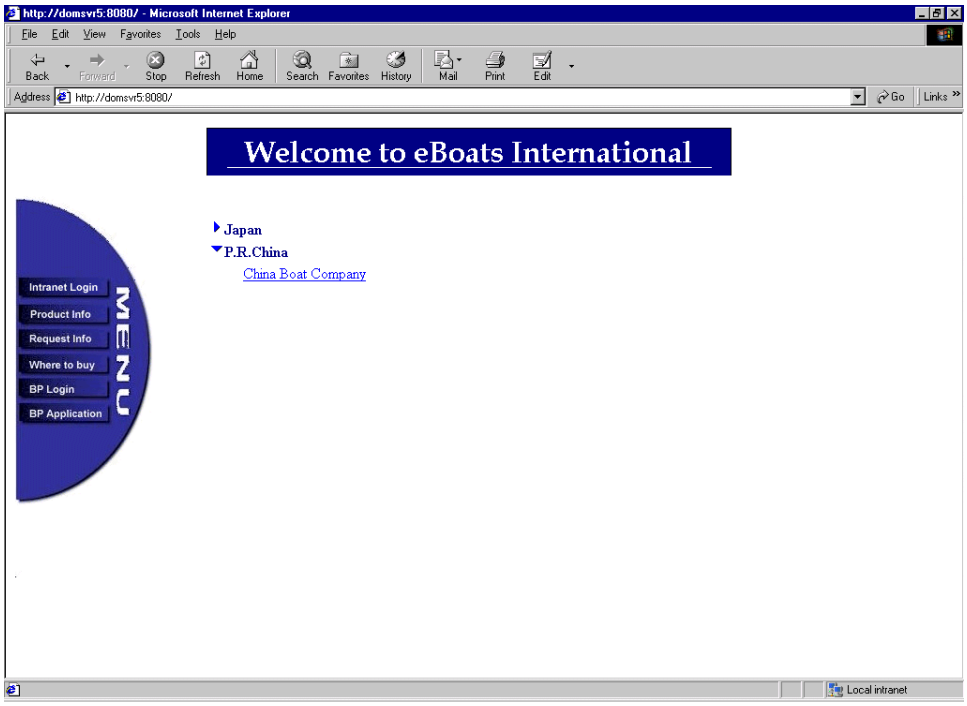


Figure 98. Asia view

The Europe view

The Europe view displays a list of all business partners in the Europe region. The view selection formula is:

```
SELECT Form="BPLocations" & Location = "Europe"
```

The view contains two columns. The first column is categorized in ascending order and shows the Country field. The second column displays the Company field. Figure 99 on page 186 shows this view.

Figure 99. Europe view

The North America view

The North America view displays a list of all business partners in the North American region. The view selection formula is:

```
SELECT Form="BPLocations" & Location = "North America"
```

The view contains two columns. The first column is categorized in ascending order and shows the Country field. The second column displays the Company field. Figure 100 shows this view.

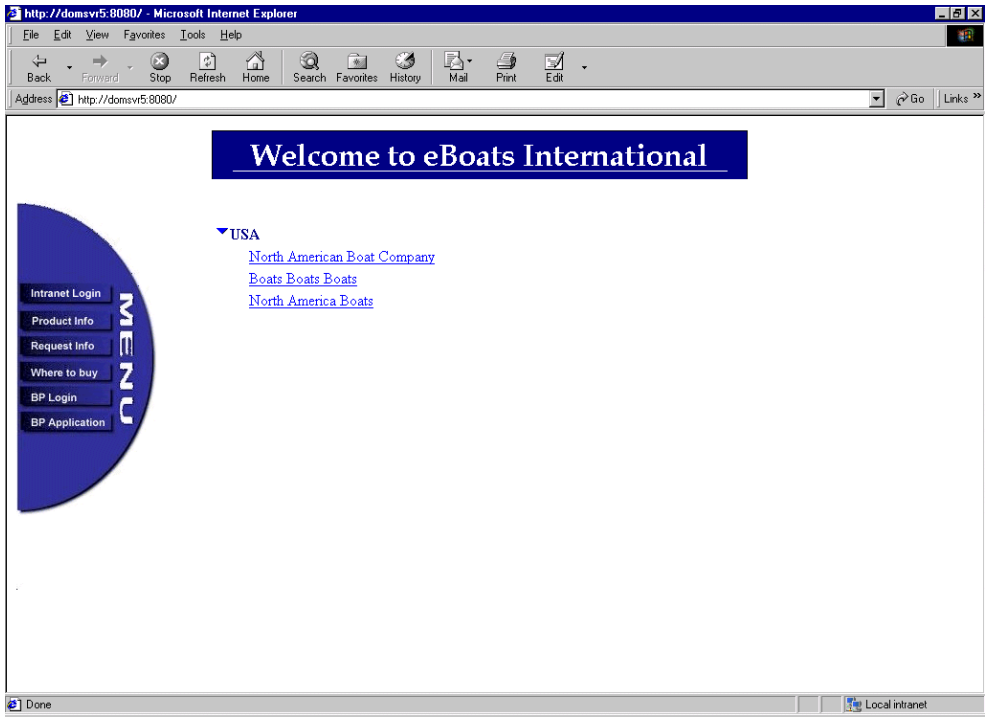


Figure 100. North America view

The Product view

The Products view displays all products. The view selection formula is:

```
SELECT Form="Product"
```

The view contains two columns. The first column is categorized in ascending order and shows the ProductLine field. The second column displays the Model field. Figure 101 on page 188 shows this view.

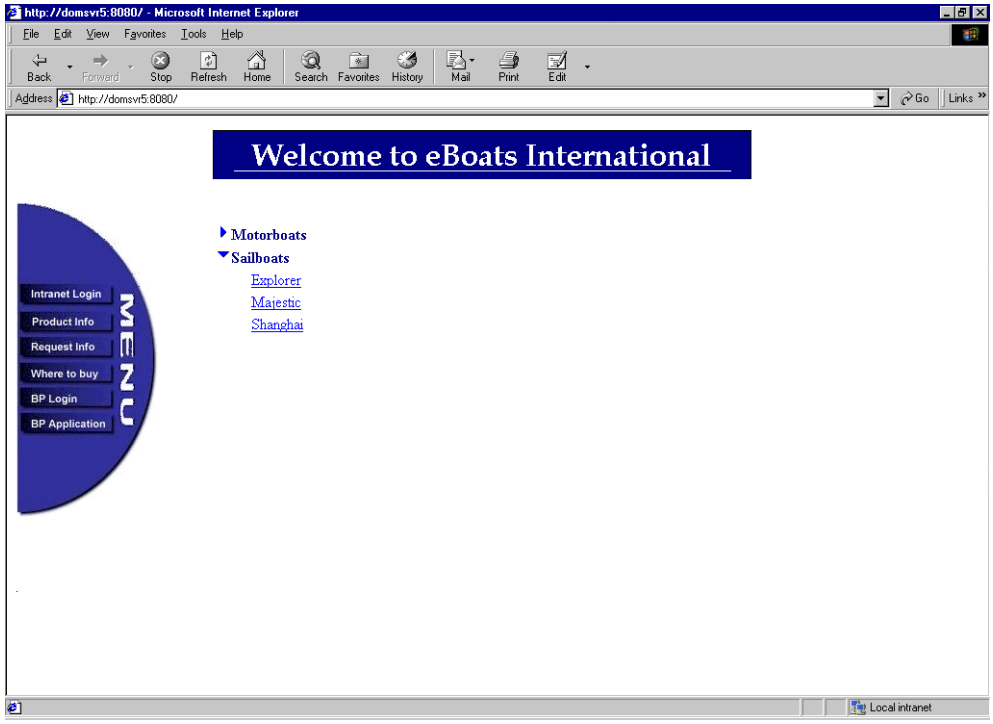


Figure 101. Products view

The AllProducts view

The AllProducts view is a hidden view that is used for product lookups from the other databases. The view selection formula is:

```
SELECT Form="Product"
```

The view contains three columns. The first column is sorted in ascending order and shows the Model field. The second column is used for lookups from the ShoppingBasket form in the eboatsB2B.nsf database. This column builds a string that contains the product model, product number, and product price. The second column contains the following formula:

```
mdl := @Left( Model ; 20 );
mdlSpace := @Repeat( " " ; 22 - @Length( mdl ) );
prodNum := @Left( ProductNumber ; 20 );
prodNumSpace := @Repeat( " " ; 22 - @Length( prodNum ) );
mdl + mdlSpace + prodNum + prodNumSpace + @Text(price)
```

The third column shows the Product Number field. Figure 102 shows this view.

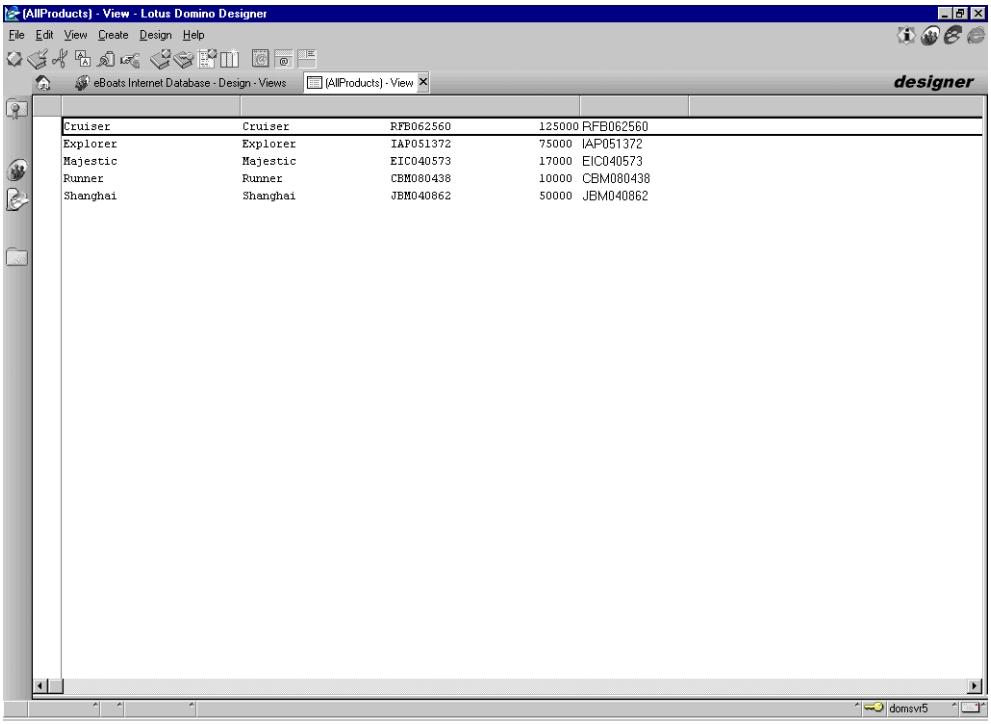


Figure 102. AllProducts view

8.2.2 Business-to-business database: eboatsB2B.nsf

When business partners login from the Web site, this is the database they will use. Access to this database is made through SSL to provide security. From this database, business partners can place orders, check an order's status, view the customer list, and check product inventory levels.

The following sections describe the database's design elements.

8.2.2.1 Security

Everyone accessing this database through the Internet has to have a valid user ID and password because in the ACL, Anonymous is set to No Access. Authorized Business Partners logon using their business partner number and password that is assigned to them when their application is approved. The group BusinessPartners, which contains every approved business partner, is listed in the ACL as Author with create document rights.

In the Advanced tab of the ACL, maximum Internet name and password access is set to Author. The appropriate administration server is selected for this database. Default Access is set to No Access.

8.2.2.2 Database launch properties

For Web users, the frameset WebFS is displayed when the database is opened. This frameset is described in 8.2.2.6, "Framesets" on page 195.

The corresponding settings are shown in the database launch properties, which can be found in Figure 103 on page 190. The tab for the database launch properties is the fifth from the left.

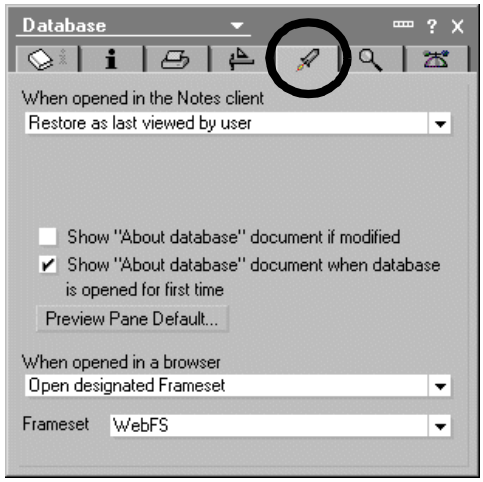


Figure 103. eboatsB2B.nsf database launch properties

8.2.2.3 Forms

This database uses the following three forms:

- BPWelcome
- Inventory Levels | InventoryQty
- Shopping Basket | ShoppingBasket

The BPWelcome form

This is the form that users see when the database first opens. Figure 104 shows the details of this form.

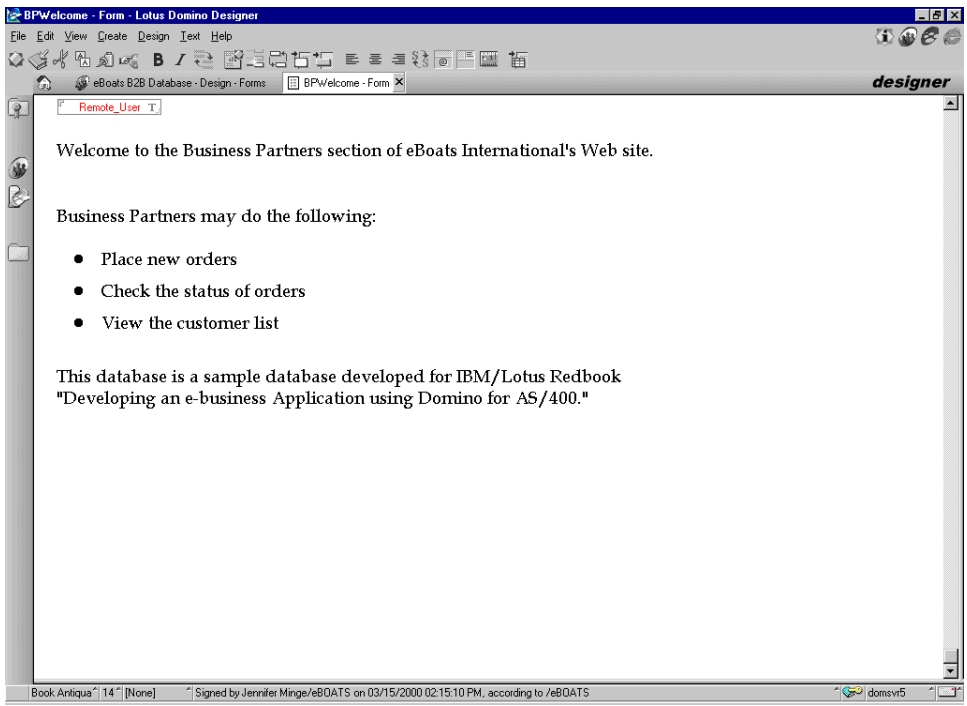


Figure 104. BPWelcome form

This form contains a single field called Remote_User. This field is used to capture the CGI variable showing the authenticated user's name. The onLoad form event

calls the JavaScript function saveBPNumber(). This function saves the user id to a cookie. This information is referenced when the business partner places an order or checks the status of orders.

The forms JSHeader contains the following JavaScript code:

```
function setCookie(name, value) {
   document.cookie = name + "=" + escape(value) + "; path=/;";
}
function saveBPNumber() {
     f = document.forms[0];
   setCookie( "BPNumber", f.Remote_User.value);
}
```

Inventory levels form

This form, displayed in Figure 105, allows a business partner to check the current inventory level for any product. This form demonstrates how to use @DbCommand, LS:DO and LSX LC. The user selects the appropriate model from the combo box, and then clicks on the appropriate button. The current inventory level for the selected model is shown in the Results field.

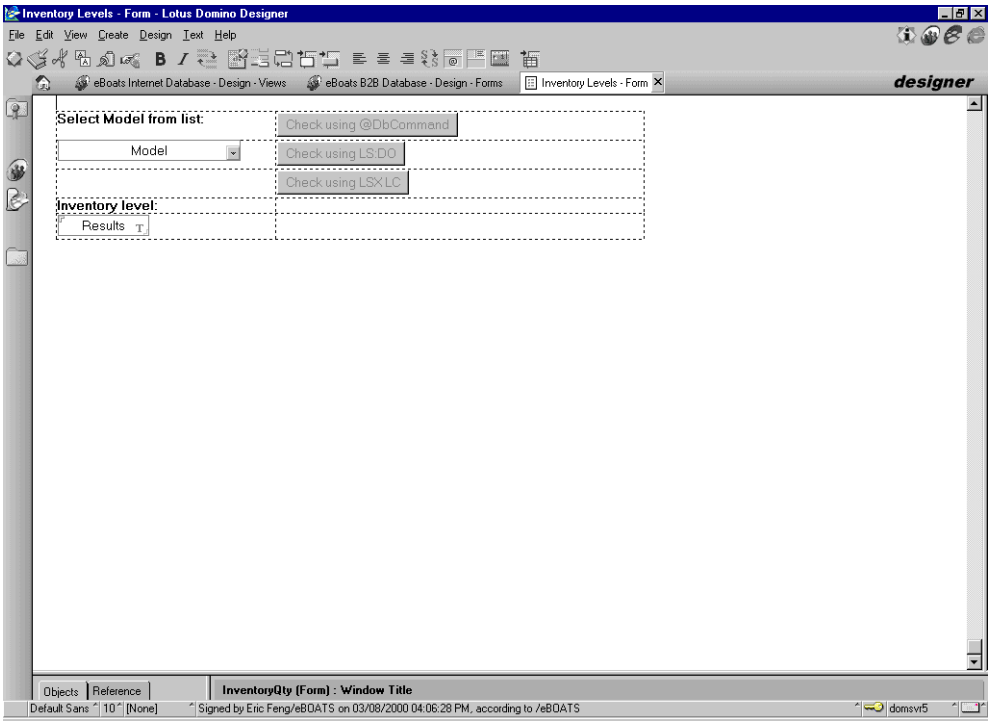


Figure 105. Inventory Levels form

The code behind the Check using @DbCommand button can be found in D.1, "@DbCommand" on page 373.

The code behind the Check using LS:DO button is:

```
@Command([ToolsRunMacro] ; "(LSDOCheckInventory)" )
```

It triggers the LSDOCheckInventory agent. The code can be found in A.3, "LS:DO CheckInventory agent" on page 330.

The code behind the Check using LSX LC button is:

```
@Command([ToolsRunMacro];"LCLSXCheckInventory")
```

It triggers the LCLSXCheckInventory agent. The code can be found in A.6, “LSX LC agent” on page 335.

Attention

During the project, we found that simultaneously using @DB functions and LS:DO on one hand and LSX LC on the other hand, in the same Domino HTTP server environment, could create a problem preventing further access to the DB2 UDB for AS/400 database.

A fix has been developed, which should be included in a future Domino for AS/400 Release 5.0 Quarterly Maintenance Release (QMR).

Shopping Basket form

This form is used to place orders. The user selects the product from the combo box, enters the quantity, and clicks the Order Now button. A cookie is used to store details of all items that are ordered. A shopping basket of ordered items is displayed dynamically using the cookie. After entering all products to be ordered, the user clicks on the Place Order button to submit the order to the ORDERINFO DB2 UDB for AS/400 table.

The shopping basket is built dynamically using passthru HTML. This code builds a table with a header row, and then loops through every entry in the cookie to display each item ordered. A grand total is printed at the end of the table. Figure 106 shows the details of the Shopping Basket form and the beginning of the passthru HTML code.

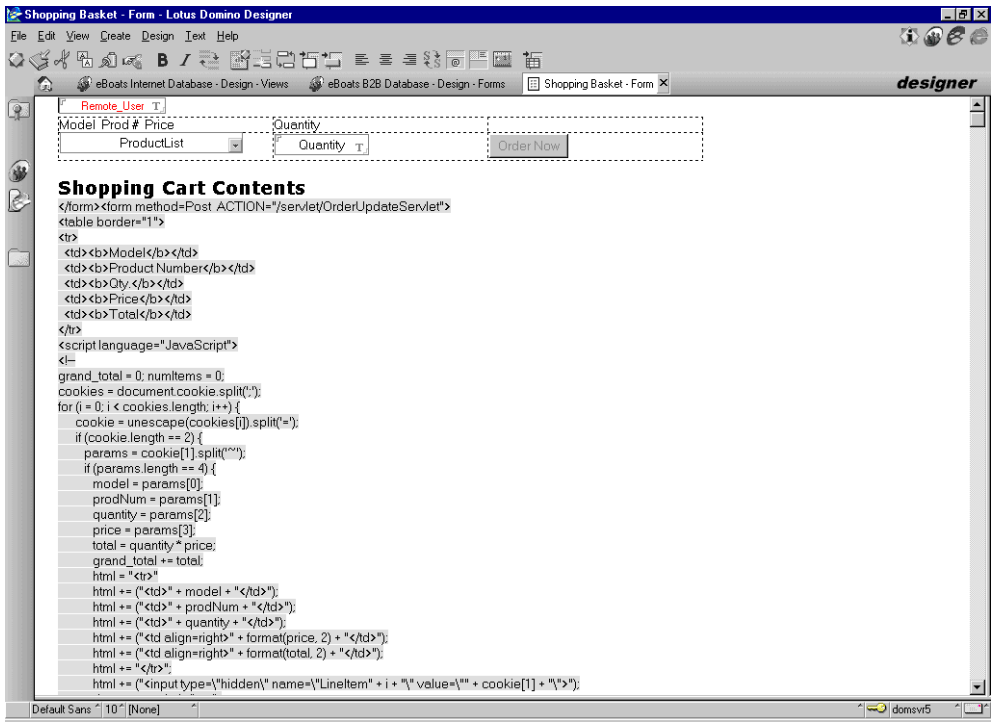


Figure 106. Shopping Basket form

The complete listing of the passthru HTML code can be found in B.2, “ShoppingBasket form passthru HTML code” on page 337.

When a user clicks the Order Now button, the JavaScript function addProduct() is called. The function adds the ordered items to the cookie, and then sets the cookie. The shopping basket is re-displayed with the updated order information. The passthru HTML code calls a JavaScript function to format the cost field as currency. If the user clicks the Clear My Shopping Cart button, a JavaScript function is called to delete the contents of the cookie.

All of the above JavaScript functions are listed in the JS Header. The complete listing of the JS Header can be found in B.3, “ShoppingBasket form JSHeader” on page 338.

8.2.2.4 Navigators

This database contains one navigator called SideNav. Figure 107 on page 194 displays the details of this navigator.

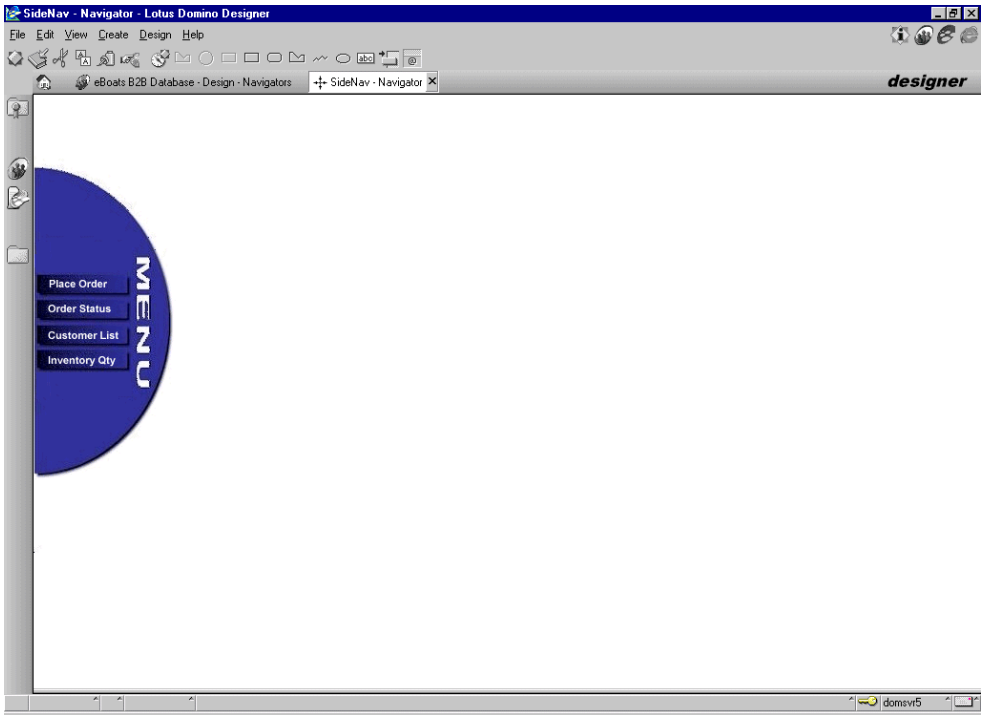


Figure 107. SideNav navigator

This navigator is displayed in the left frame of the WebFS frameset. From this Navigator, Business Partners can place orders, check order status, view the customer list, and check inventory levels. Each of these options is described in the following sections.

When a user clicks Place Order, the ShoppingBasket form, which is found in the current Domino database, is displayed. The code is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/ShoppingBasket?OpenForm";
@URLOpen(url)
```

When a user clicks Order Status in the navigator, a Java servlet on the AS/400 system is called, using the following code:

```
db := @Subset(@DbName; -1);
url := "/servlet/OrderQueryServlet";
@URLOpen(url)
```

The Java servlet name is OrderQueryServlet. This servlet reads the cookie to get the Business Partner's number, and then it passes this information to a JavaBean called OrderInfoQuery. This JavaBean then calls a stored procedure called QueryOrder. The stored procedure looks up all orders in the ORDERINFO DB2 UDB for AS/400 table based on the business partner number. The stored procedure returns a result set to the JavaBean. The JavaBean passes the result set to the Java servlet. The Java servlet loops through the result set and displays all orders to the business partner.

The code for the OrderQueryServlet Java Servlet can be found in C.1.1, "OrderQueryServlet servlet" on page 343.

The code for the OrderInfoQuery JavaBean can be found in C.2.1, “OrderInfoQuery JavaBean” on page 350.

The code for the queryOrder stored procedure can be found in D.2, “Stored procedure queryOrder” on page 373.

When a user clicks Customer List in the navigator, the view CustomerList in the eboatsCustomer.nsf database is displayed. The eboatsCustomer.nsf database contains information on all customers who have requested information from the Web site. Business partners are given reader access to this database so they may view eBoats International’s customer list. The code to display this view is:

```
@URLOpen( "/eboatscustomer.nsf/CustomerList?OpenView" )
```

When a user clicks Inventory Qty, the InventoryQty form is used. The code is:

```
@Command([Compose] ;  "InventoryQty")
```

8.2.2.5 Pages

This database contains one page called WelcomeBanner. This page is displayed at the top of the WebFS frameset. Figure 108 shows the details of this page.

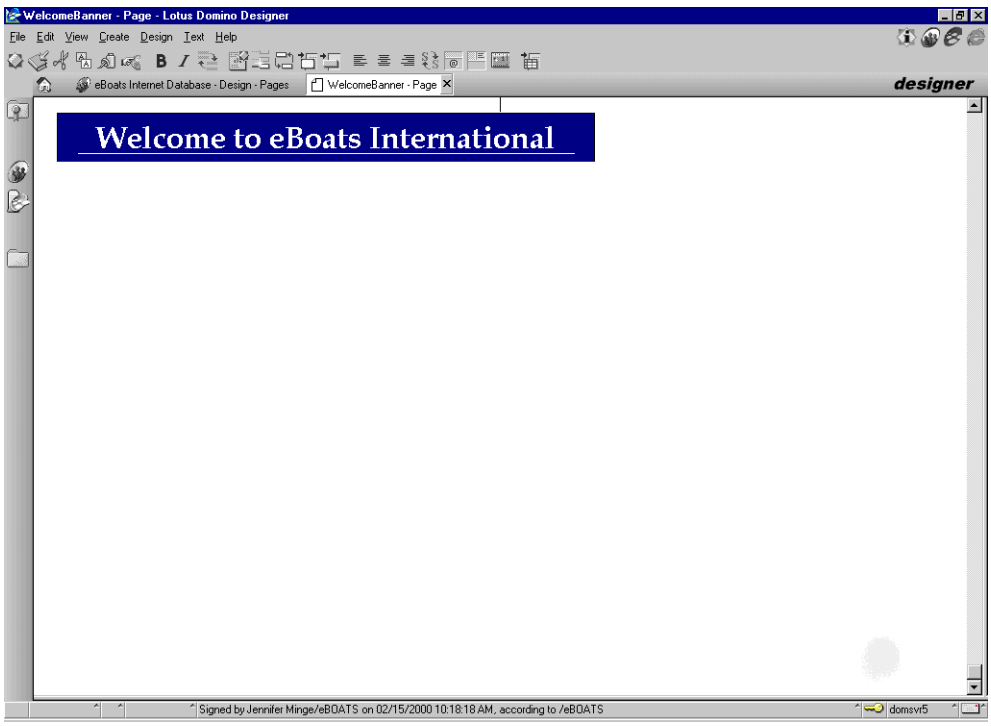


Figure 108. WelcomeBanner page

8.2.2.6 Framesets

This database contains one frameset called WebFS. This frameset has three frames. The left frame displays the SideNav navigator, the top frame displays the WelcomeBanner page, and the right frame displays what is selected from the SideNav navigator. By default the right frame displays the BPWelcome form. Figure 109 on page 196 shows the details of this frameset.

This frameset is used whenever opening the eboatsB2B.nsf Domino database, as described in 8.2.2.2, “Database launch properties” on page 189.

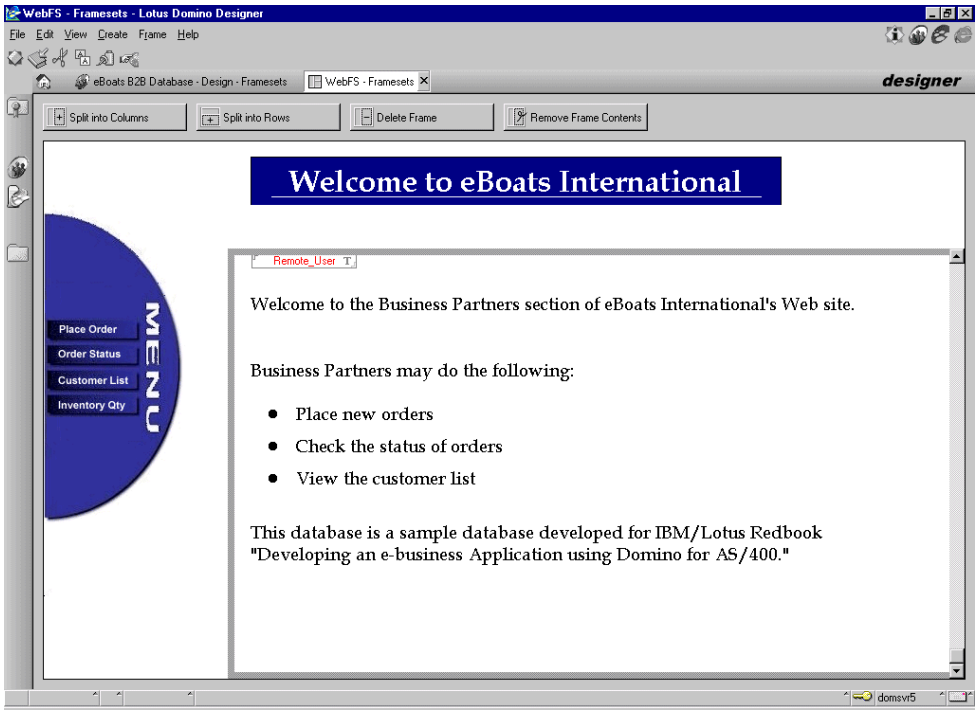


Figure 109. WebFS frameset

8.2.3 Intranet database: eboatsIntranet.nsf

When eBoats International employees login from the Web site, this is the database they will use. Access to this database is made through SSL to provide security. From this database, employees can check the status of all orders or just the orders for a particular business partner. The order status information is displayed using XML.

8.2.3.1 Security

Everyone accessing this database through the Internet has to have a valid user ID and password because in the ACL entry, Anonymous is set to No Access. eBoats International employees access this database using their user name and Internet password, which are specified in the Domino Directory. The group BusinessPartners, which contains every approved business partner, is listed in the ACL as No Access. The appropriate access levels are given to eBoats International employees.

In the Advanced tab of the ACL, maximum Internet name and password access is set to Author. The appropriate administration server is selected for this database. Default Access is set to Author.

8.2.3.2 Database launch properties

For Web users, the frameset WebFS is displayed when the database is opened. This frameset is described in 8.2.3.7, “Framesets” on page 205.

The corresponding settings are shown in the database launch properties, which can be found in Figure 110. The tab for the database launch properties is the fifth from the left.

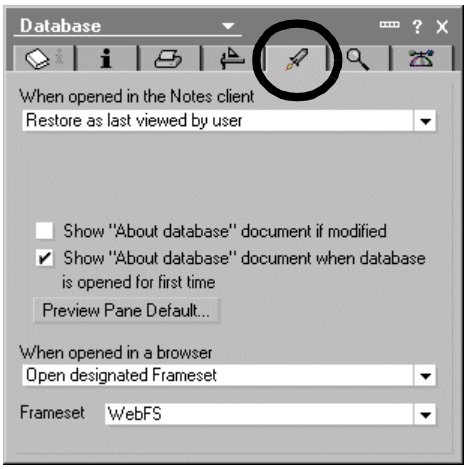


Figure 110. eboatsIntranet.nsf database launch properties

8.2.3.3 Forms

This database uses the following three forms:

- Order
- GetBPNumber
- ProfileDoc
- Stylesheet

The Order form

This form contains all of the fields displayed for order status. The form is formatted with XML formatting. In addition, the form properties for Web access is set to treat document contents as HTML. Figure 111 shows the details of this form.

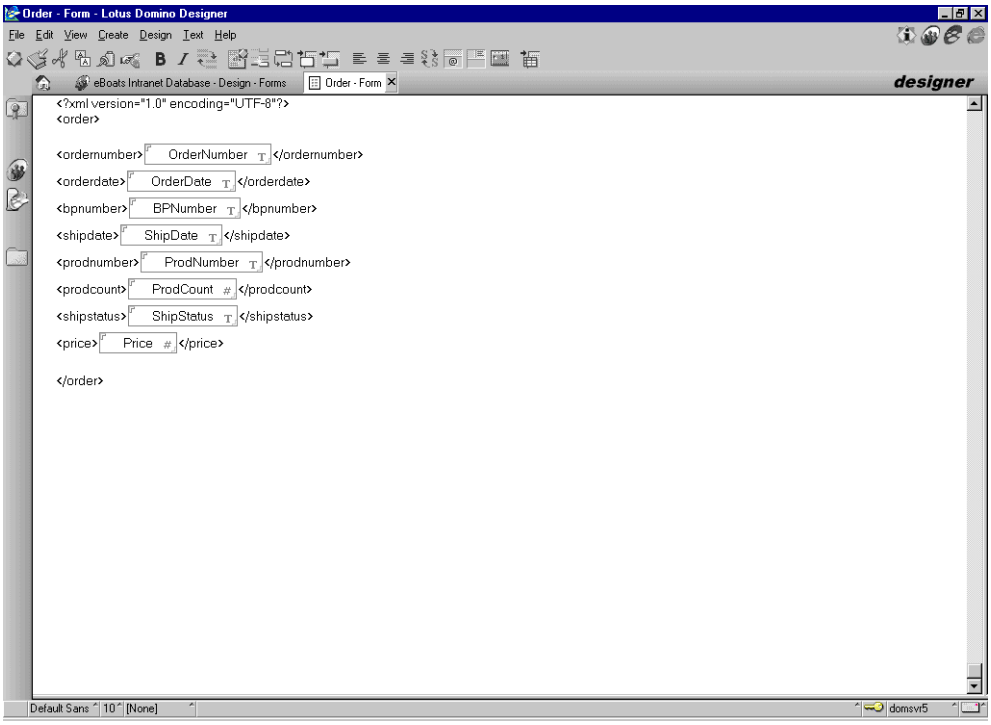


Figure 111. Order form

The first line of the form is the XML header showing that we are using Version 1.0 of XML. The second and last line form the parent tag called order. The remaining lines are child tags, which describe the data to be displayed.

On an hourly basis, LEI does a replication from the ORDERINFO DB2 UDB for AS/400 table to this database. This activity copies the contents of all fields. Figure 112 shows the LEI replication activity.

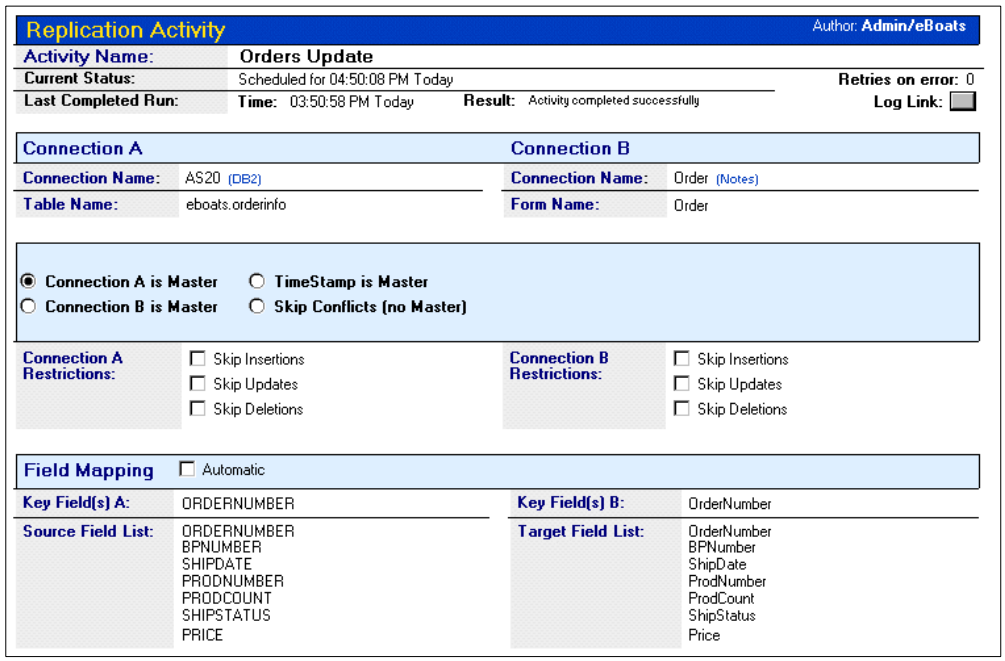


Figure 112. LEI direct transfer activity

This replication activity uses two connection documents called AS20 and Order. Figure 83 on page 174 shows the details of AS20 connection document. Figure 113 shows the details of the Order connection document. The activity is scheduled to run every hour.

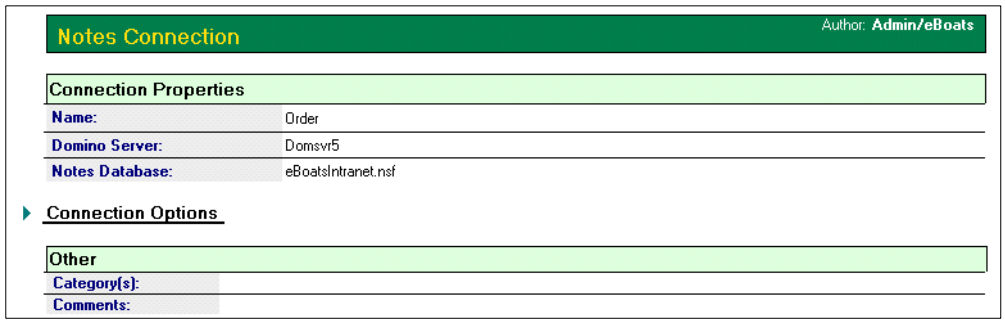


Figure 113. LEI order connection document

An activity report is reported each time for the activity. An example is shown in Figure 114. As can be seen in the figure, two new orders were recorded in the AS/400 database since the previous time the activity ran and were successfully inserted in the Domino database.

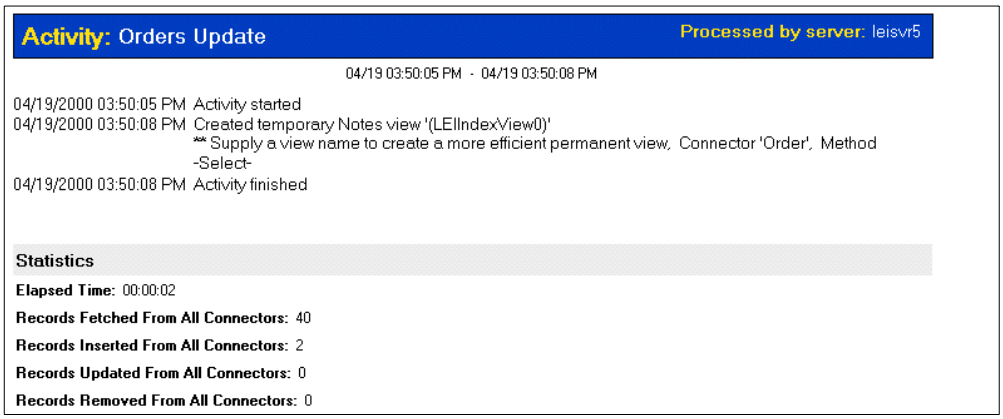


Figure 114. Orders Update activity report

The GetBPNumber form

eBoats International employees have the option of displaying all orders for selected business partners. From this form, the user selects the business partner number that they want displayed. Figure 115 displays the details of this form.

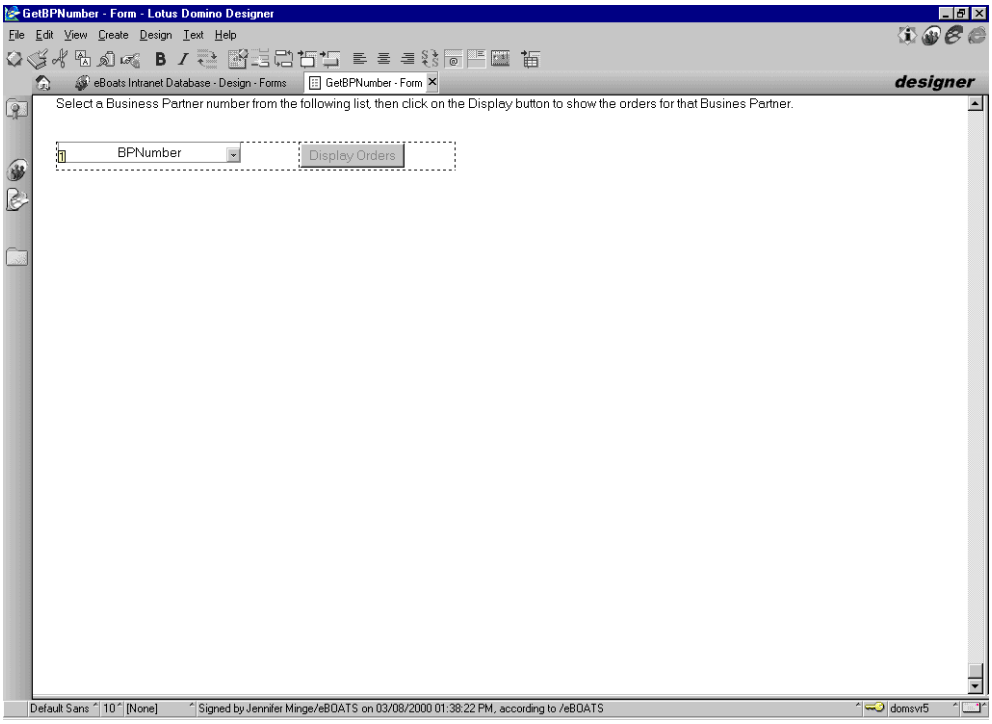


Figure 115. GetBPNumber form

The user selects the BPNumber from the combo box. The combo box uses the following formula to get a list of all business partner numbers from the ORDERINFO DB2 UDB for AS/400 table:

```
@DbColumn( "ODBC" : "NoCache" ; "AS20" ; "USERID" ; "PASSWORD" ;
"EBOATS.ORDERINFO" ; "BPNUMBER"  )
```

After selecting the appropriate business partner number, the user clicks the Display Orders button. The code for this button is:

```
@SetProfileField( "ProfileDoc" ; "BPNumber" ;  BPNumber );
@Command([ToolsRunMacro] ; "(SelectedBP)" )
```

This button sets a field on a profile document to the selected business partner number, and then it calls the agent “(SelectedBP)”. The code for this agent is listed in A.4, “SelectedBP agent” on page 332.

This agent connects to the ORDERINFO DB2 UDB for AS/400 table using LS:DO, executes an SQL select statement, and gets a result set of all orders for the selected Business Partner number. The agent loops through this result set and prints XML commands to the display to show the formatted data. The data is formatted using the ordercatalog.xsl style sheet, which is described in D.3, “orderCatalog.xsl Page” on page 395.

The ProfileDoc form

This form contains a single field called BPNumber. This field contains the selected business partner number, and the LS:DO script will access this form to get the number. Figure 116 shows the details of this form.

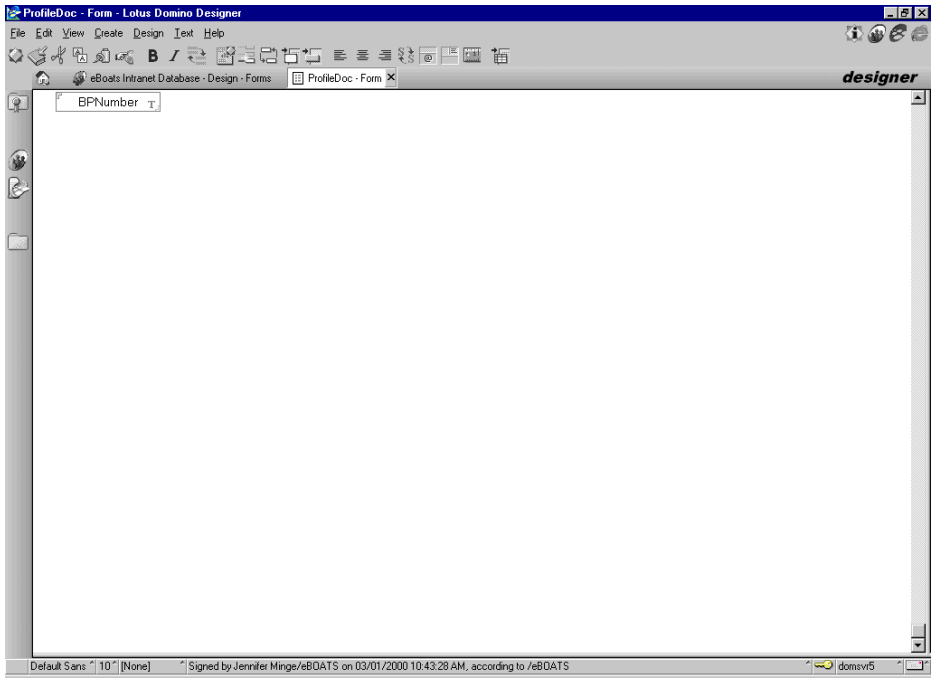


Figure 116. ProfileDoc form

The Stylesheet form

This form contains two fields:

- **StylesheetName**: This field contains the XSL style sheet name.
- **HTML**: This field contains the contents of the style sheet.

Figure 117 shows the details of this form.

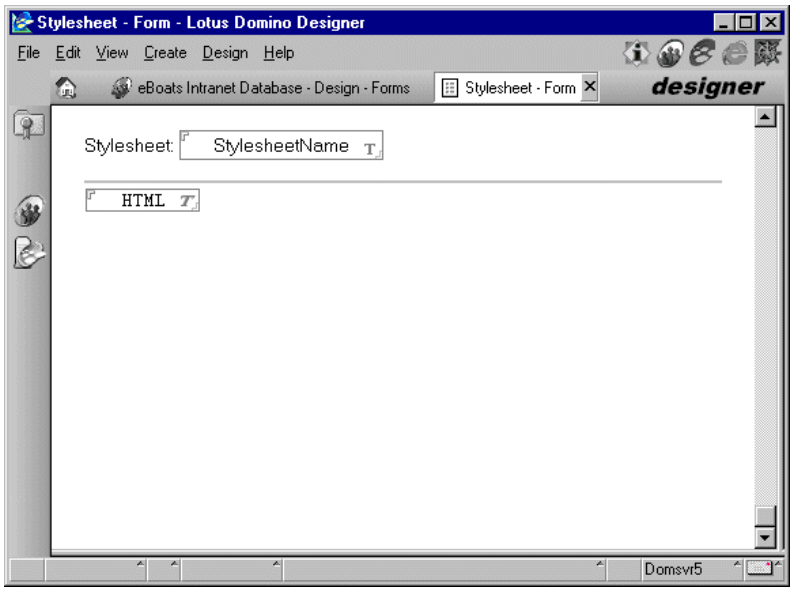


Figure 117. Stylesheet form

8.2.3.4 Views

This database contains a single view called Order. The view selection formula is:

```
SELECT Form="Order"
```

The view attributes are set to treat contents as HTML for Web access. The view has eight columns, with one column for each field displayed on the Order form. Every column displays a field surrounded by XML tags. The first and last column correspond to the first and last entry defined in the ordercatalog.xsl style sheet, which is described in D.3, “orderCatalog.xsl page” on page 373.

The code contained in the first column is:

```
"<order><ordernumber>"+OrderNumber+"</ordernumber>"
```

The code contained in the second column is:

```
"<orderdate>"+OrderDate+"</orderdate>"
```

The code contained in the last column is:

```
"<price>"+@Text(Price)+"</price></order>"
```

8.2.3.5 Pages

This database contains the following four pages:

- WelcomeBanner
- Welcome
- orderCatalog.xsl
- OrderStyleSheet

WelcomeBanner page

This page is displayed at the top of the WebFS frameset. Figure 118 on page 202 shows the details of this page.

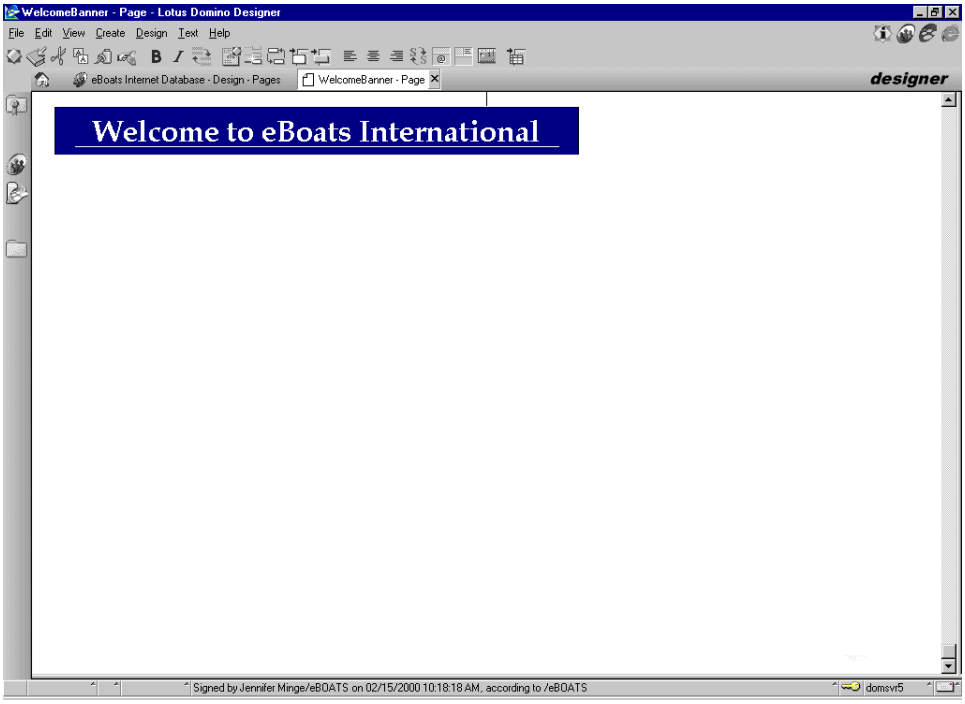


Figure 118. WelcomeBanner page

Welcome page

This page, displayed when the database first opens, contains information about the database and the technologies that are demonstrated in the database. This page is displayed in the WebFS frameset when the database is launched. Figure 119 shows the details of this page.

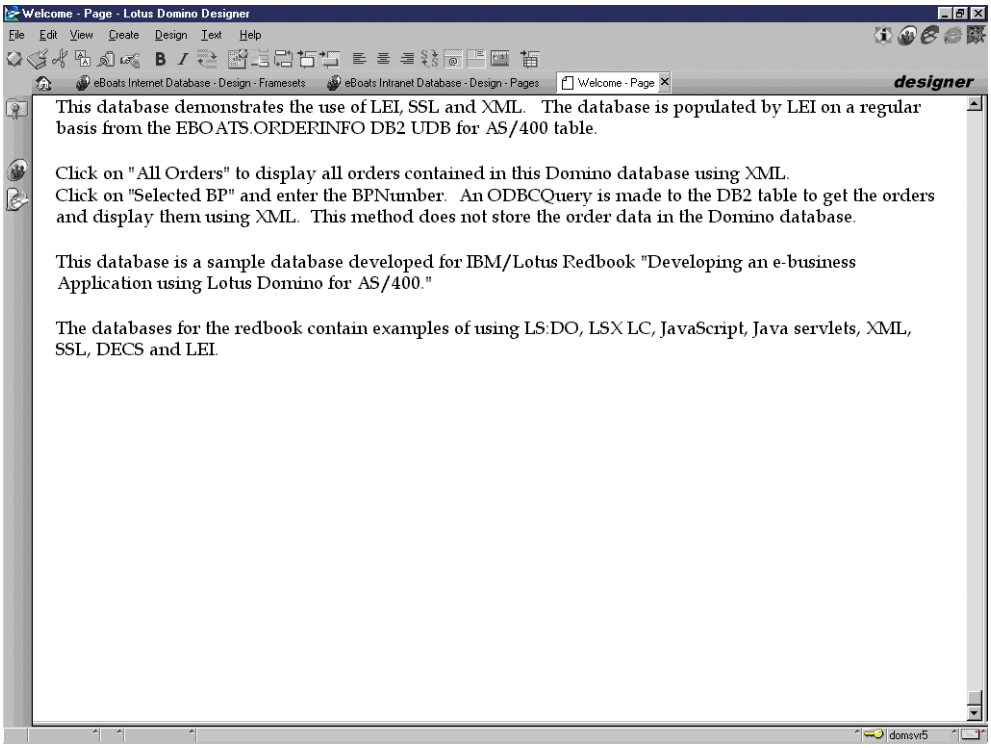


Figure 119. Welcome page

orderCatalog.xsl page

This page contains the XSL style sheet used to format the XML data. This style sheet creates a table header row, then loops through every item contained in the Order view, and displays the information. The code for this page is listed in D.3, “orderCatalog.xsl page” on page 373.

OrderStyleSheet page

This page contains XML formatting code with an embedded view showing the Order view. The XML formatting tells it to use the ordercatalog.xsl style sheet to format the data. Figure 120 on page 204 shows the details for this page.

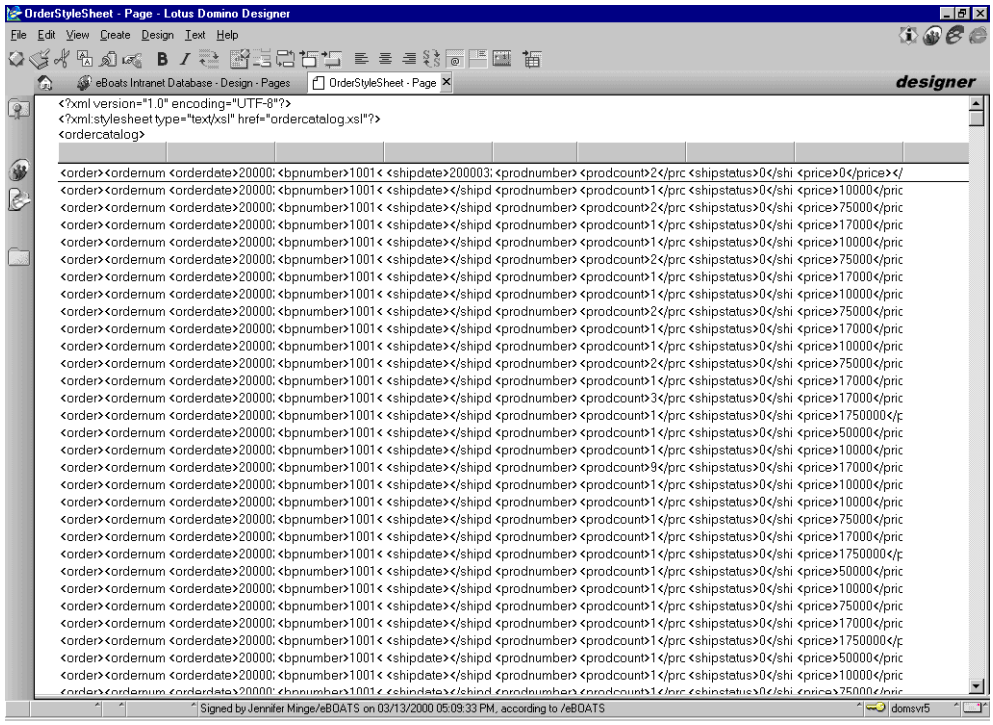


Figure 120. OrderStyleSheet page

8.2.3.6 Navigators

This database contains one navigator called SideNav. Figure 121 displays the details of this navigator.

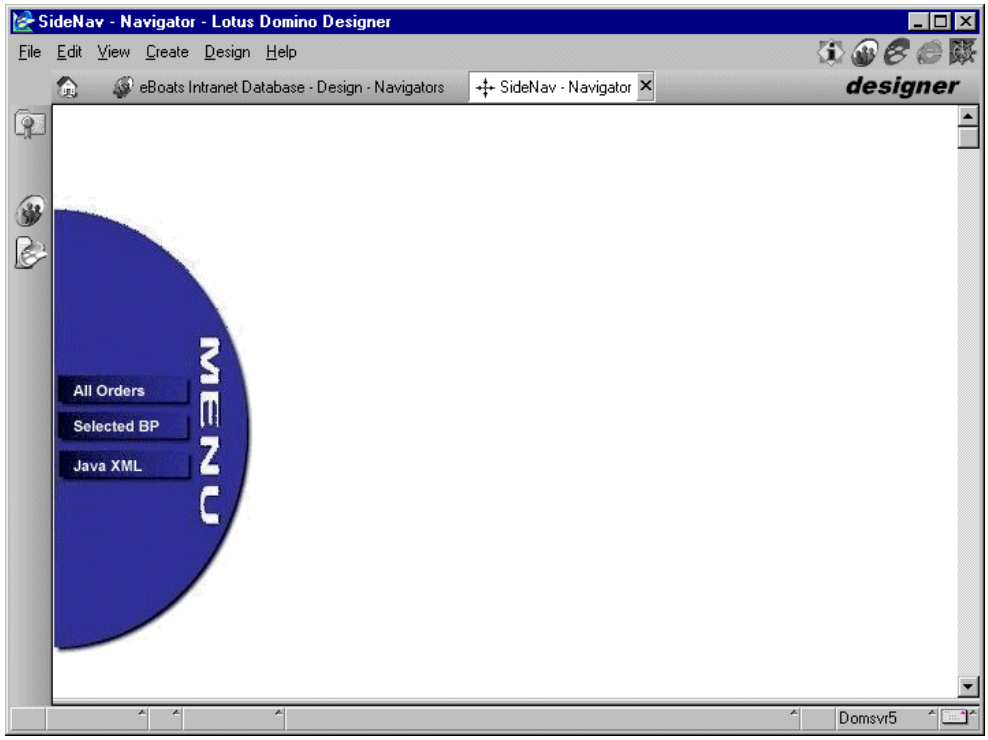


Figure 121. SideNav navigator

This navigator is displayed in the left frame of the WebFS frameset. From this navigator, employees can check the status for all orders or just the orders for a specified business partner. Each of these options is described in the following sections.

All Orders
When a user clicks All Orders, the OrderStyleSheet page is displayed. This displays a list of all orders that are stored in this database. Information is displayed using XML. Orders are transferred from the ORDERINFO DB2 UDB for AS/400 table using LEI, as described in 8.2.3.3, “Forms” on page 197.

The code is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/OrderStyleSheet?OpenPage";
@URLOpen(url)
```

Selected BP
When a user clicks Selected BP, the GetBPNumber form is displayed. This displays orders for a selected business partner without storing the order information in the Notes database. Information is displayed using XML.

The code is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/GetBPNumber?OpenForm";
@URLOpen(url)
```

Java XML
When a user clicks Java XML, the orderXML Java agent is called. This displays all the orders without storing the order information in the Notes database. Information is displayed using XML.

The code is:

```
db := @Subset(@DbName; -1);
url := "/" + db + "/orderXML?OpenAgent";
@URLOpen(url)
```

The code of the Java agent orderXML is in C.4.1, “orderXML” on page 361.

8.2.3.7 Framesets
This database contains one frameset called WebFS. This frameset has three frames. The left frame displays the SideNav navigator, the top frame displays the WelcomeBanner page, and the right frame displays what is selected from the SideNav navigator. By default, the right frame displays the Welcome page.

This frameset is used whenever opening the eboatsIntranet.nsf Domino database, as described in 8.2.3.2, “Database launch properties” on page 196.

Figure 122 on page 206 shows the details of this frameset.

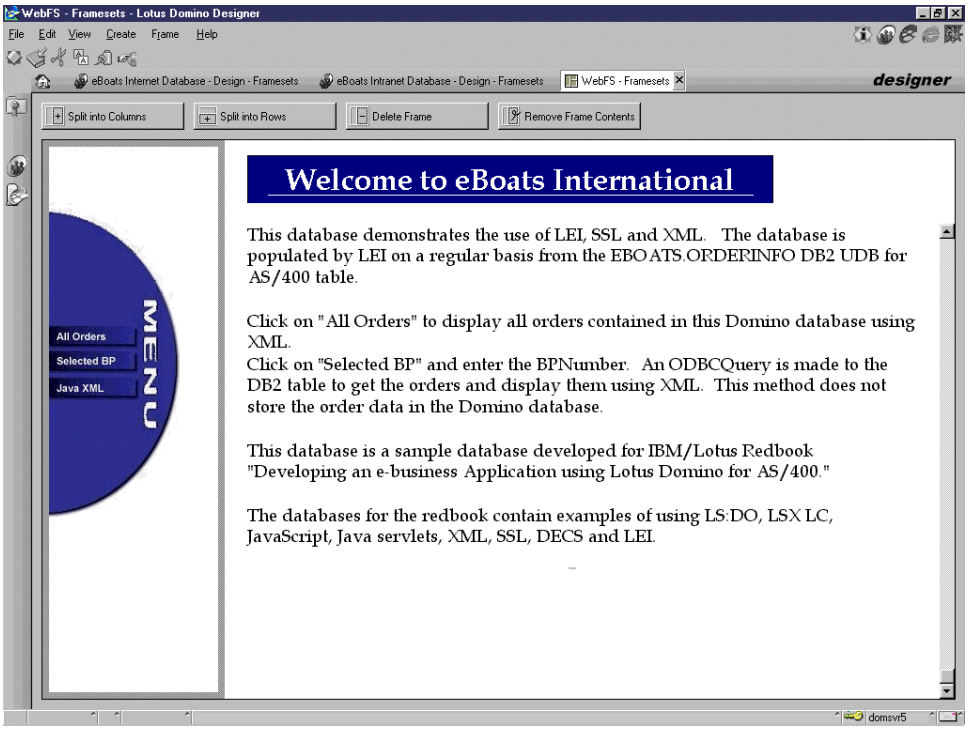


Figure 122. WebFS frameset

8.2.4 Business partner database: eboatsbp.nsf

All business partner applications are created in this database. The Notes workflow capabilities are used to route new applications to the designated person or persons responsible for approving or rejecting the application.

The following sections describe the database's design elements.

8.2.4.1 Security

Default Access is set to Depositor. This allows users to submit the application online. The appropriate access levels are given to eBoats International employees who will be responsible for approving or rejecting the application.

In the Advanced tab of the ACL, maximum Internet name and password access is set to Depositor. The appropriate administration server is selected for this database.

8.2.4.2 Forms

This database uses the following three forms:

- Business Partner Application | BP
- Approver | Approver
- Registration Profile | RegistrationProfile

Business Partner Application form

The Business Partner (BP) form is filled out by any company that is interested in becoming an eBoats International business partner. Users are required to enter information into all fields that are red. JavaScript is used in the form in the OnLoad event, JSHeader, HTML Attributes and onFocus events for most fields,

and for the onClick event for the Submit button. Figure 123 shows the details for this form.

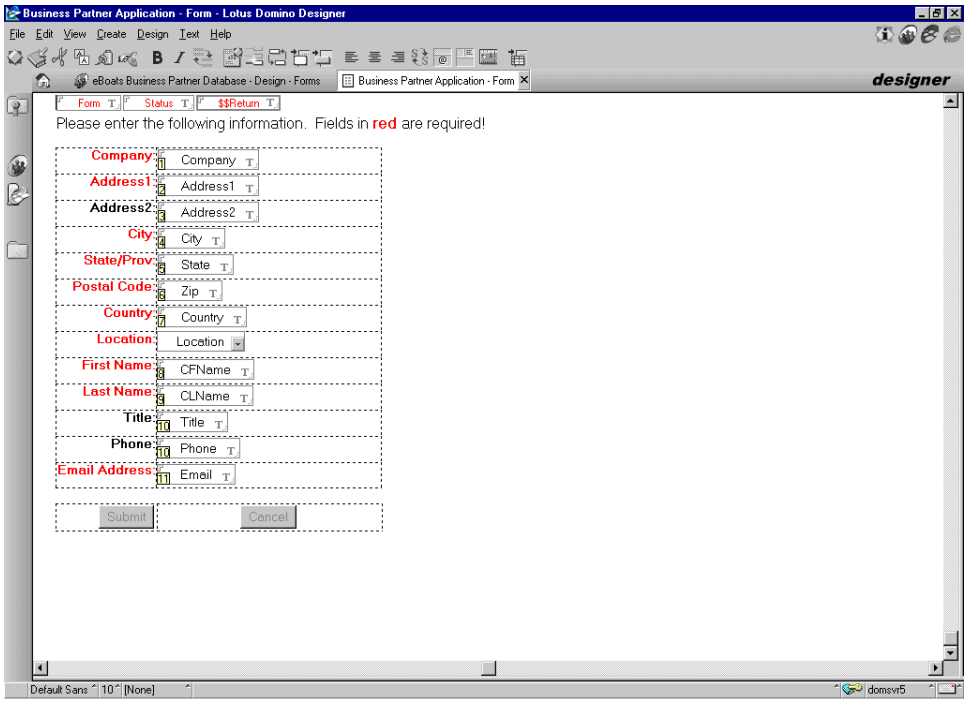


Figure 123. Business Partner Application form

The user enters the required information into the fields and then clicks the Submit button. The form onSubmit event calls the validate() function. This JavaScript function validates that all required fields contain information and that the length of the data in the fields is not longer than what is supported in the BPINFO DB2 UDB for AS/400 table. The JavaScript validate() function is listed in B.4, “BP form validate function” on page 339.

After the form has been validated, the agent “NewBPNotification” is called from the WebQuerySave event. The code for the LotusScript agent is listed in A.1, “NewBPNotification agent” on page 327.

Note

This form is also used in an example in chapter “FxD in action” of the redbook *Lotus Fax for Domino for AS/400: Getting the Straight Facts*, SG24-5941. The example demonstrates that fax can be part of an e-business application.

The redbook shows how to set up, configure, use and manage Lotus Fax for Domino for AS/400.

When an application is approved, the agent “(RegisterBP)” is called. The code for the LotusScript agent is listed in A.2, “RegisterBP agent” on page 328.

onLoad event

This event is used to put the cursor in the Company name field when the form is first loaded. Here is the JavaScript code:

```
var f=document.forms[0];
f.Company.focus();
```

HTML Attributes

The HTML Attributes field event is used to attach HTML to a field. We use it to define the SIZE for text fields or the number of ROWS and COLS for rich text fields. The HTML code must be enclosed in quotes. Here is the code for the Company field:

```
"SIZE=30"
```

onFocus event

The onFocus event is executed when the user enters a field. This event is used to provide field help. Here is the code for the Company field:

```
window.status="Please enter your Company Name"
```

onClick event of the Submit button

This event is responsible for validating the form before it is submitted to the Domino Server. This event calls a JavaScript function called validate. Here is the JavaScript code:

```
validate()
```

JSHeader

The JSHeader contains the validate() function which is available to every event in this form. The validate function verifies that all mandatory fields contains information and that the information in the fields does not exceed the maximum length in the BPINFO DB2 UDB for AS/400 table. The JavaScript code for the validate function is listed in B.4, “BP form validate function” on page 339.

After an application is submitted, an eBoats International employee is responsible for approving or rejecting the application. An approve button is shown only to Notes users. When an application is approved the following code is executed:

```
Set workspace = New NotesUIWorkspace
Set uidoc = workspace.currentDocument

Set session = New  NotesSession
Set db = session.CurrentDatabase
Set Agent = db.GetAgent("(RegisterBP)")
Set doc = uidoc.document
ParamDocID = doc.NoteID
Call Agent.RunOnServer(ParamDocID)
Call uidoc.close()
```

The code for the agent “(RegisterBP)” is listed in A.2, “RegisterBP agent” on page 328.

Approver form

This form contains the name or names of the people who will be notified when a business partner application is submitted online. Figure 124 shows the details for this form.

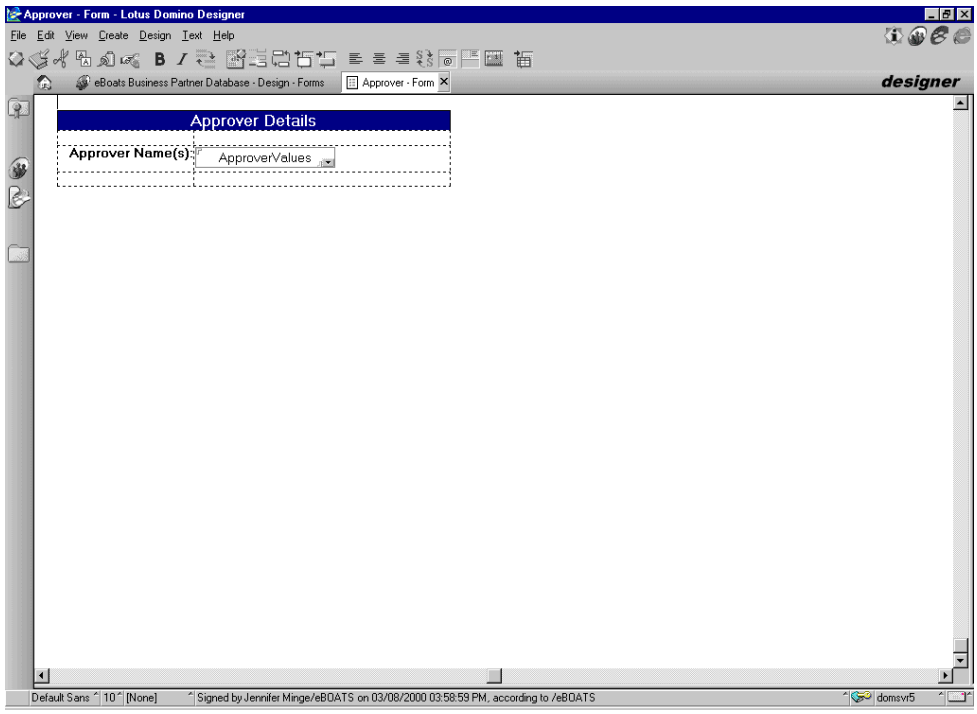


Figure 124. Approver form

Registration Profile form

This profile form helps store the name of the secondary Domino Directory that is created to contain all registered business partners, the name of the group that contains all registered business partners, and the name of the primary Domino Directory where this group is created. When a business partner application is approved, the RegisterBP agent will lookup the values in the profile document, as explained in 8.2.4, “Business partner database: eboatsbp.nsf” on page 206, and 10.1.3, “Business partners” on page 238.

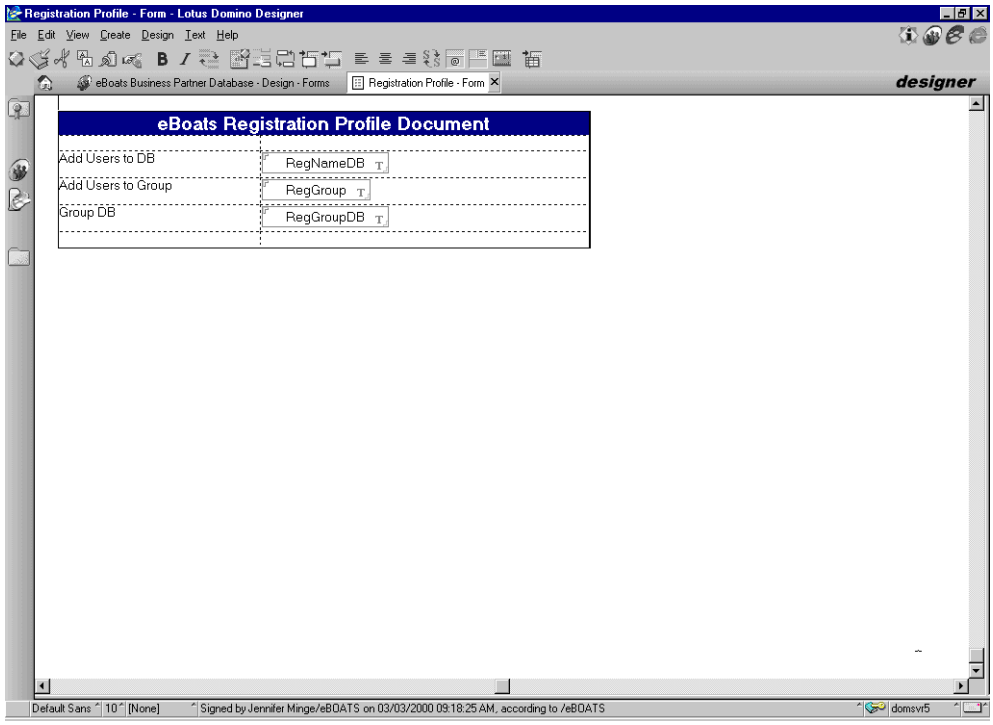


Figure 125. Registration Profile form

Note

Profile forms are useful for collecting user-specific or database-specific values.

These values are stored in Profile documents. You can use and form to create a profile document. After creating the form, you create a button, action, or agent for the application that uses either @CommandEditProfile in a formula or UIWorkspace.EditProfile or NotesDatabaseGetProfileDocument in a LotusScript program to create or retrieve a document.

In each case, Notes looks for a profile document with the form name you specify and creates a profile document if one does not already exist.

When an application is approved the RegisterBP agent will lookup the values on this form to use in registering the new Business Partner.

8.2.5 Customer database: eboatsCustomer.nsf

All information requests are created in this database. Notes workflow capabilities can be used to route the request to the nearest Business Partner who would answer the request.

The following sections describe the database's design elements.

8.2.5.1 Security

Default Access is set to Depositor. This allows users to submit the information request online. The appropriate access levels are given to eBoats International employees.

In the Advanced tab of the ACL, maximum Internet name and password access is set to Depositor. The appropriate administration server is selected for this database.

8.2.5.2 Forms

This database uses the following two forms:

- Customer Info Request | Customer
- $$ViewTemplateDefault

Customer Info Request form

This is the form customers fill out when they request information online. The user enters the required information into the fields, and then clicks the Submit button. The button calls the function validate(). This JavaScript function validates that all required fields contain information and that the length of the data in the fields is not longer than what is supported in the CUSTOMER DB2 UDB for AS/400 table. The JavaScript function validate() is listed in B.5, “Customer Info Request form validate function” on page 340.

After the form has been validated, the agent “(RegisterCustomer)” is called from the WebQuerySave event. The code for the LotusScript agent is listed in A.5, “RegisterCustomer agent” on page 333.

On an hourly basis, LEI does a direct transfer from the CUSTOMER DB2 UDB for AS/400 table to this database. This activity only transfers the contents of the CustNum, FName, LName, and Country fields. Only these fields are stored in the Domino database because they are used in a view. The remaining fields are retrieved with a DECS real-time activity when a Business Partner opens one of the customer forms. Figure 126 shows the LEI direct transfer activity.

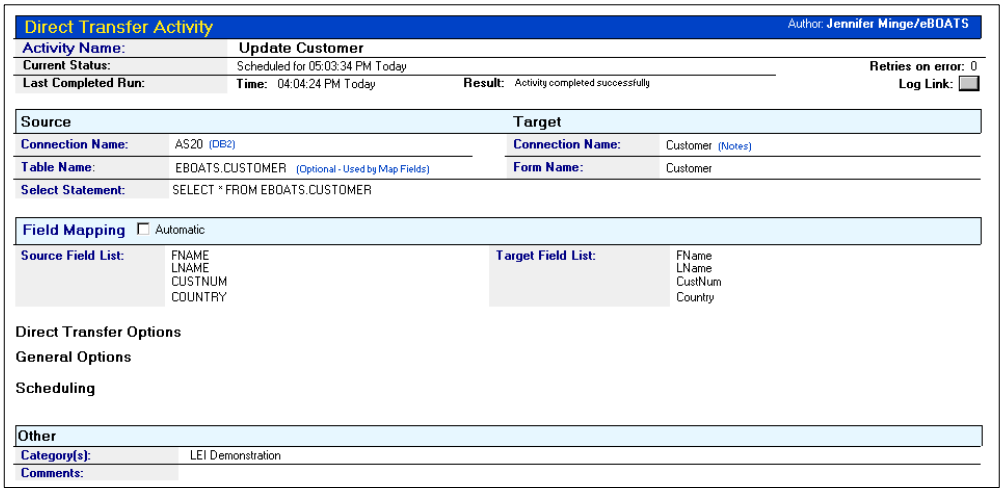


Figure 126. LEI direct transfer activity

This direct transfer activity uses two connection documents called AS20 and Customer. Figure 83 on page 174 shows the details of AS20 connection document. Figure 127 on page 212 shows the details of the Customer connection document.

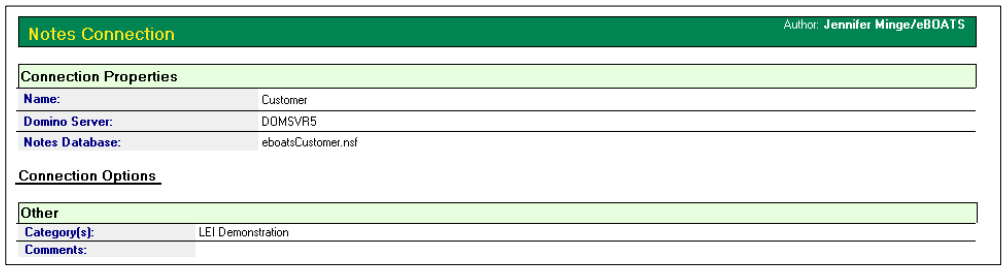


Figure 127. LEI customer connection document

All other fields are displayed using a DECS RealTime activity when the user opens the form. Figure 128 shows the DECS RealTime activity.

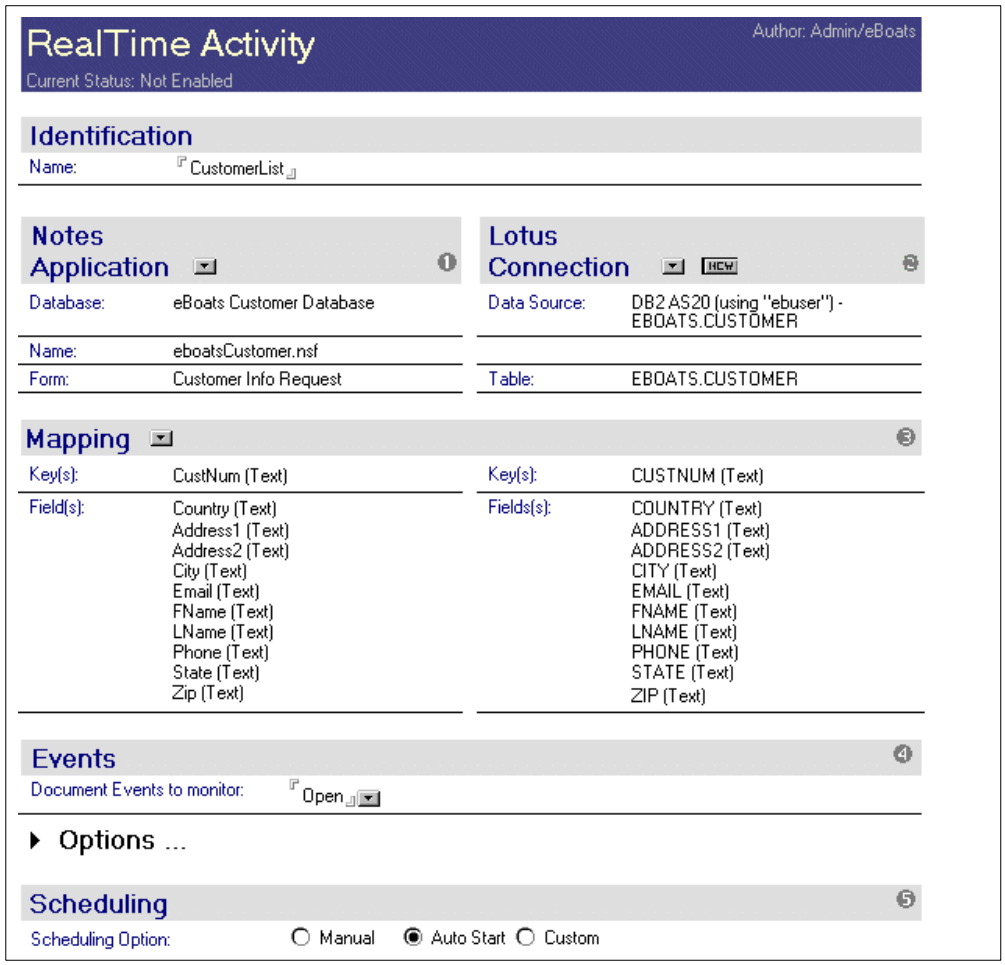


Figure 128. CustomerList DECS RealTime activity

This RealTime activity uses one connection document called automatically by DECS EBOATS.CUSTOMER. Figure 129 shows the details of AS20 connection document.

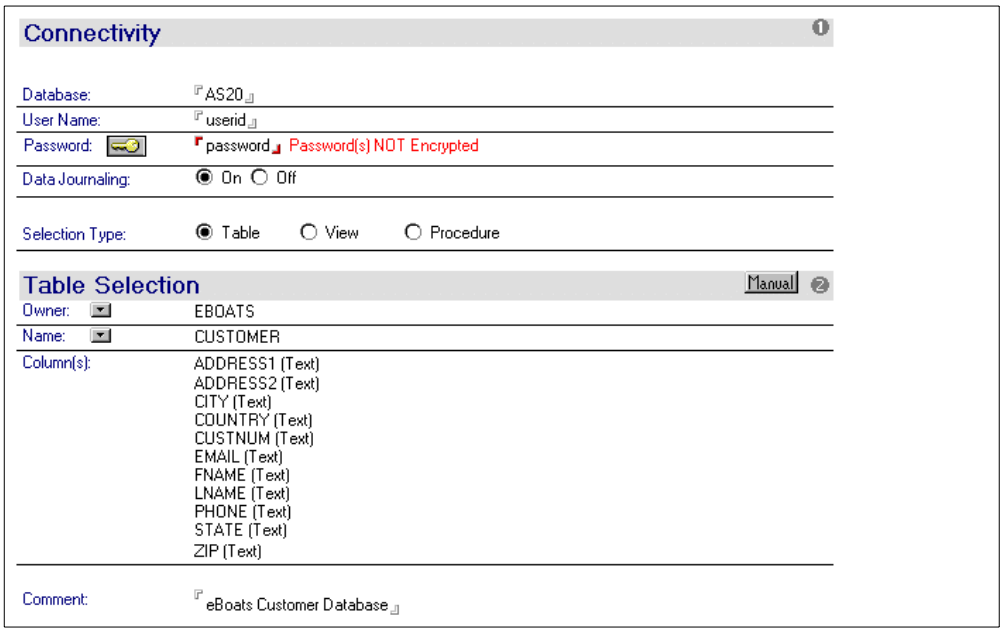


Figure 129. DECS customer connection document

Chapter 9. Sample project Domino server setup

This chapter's objective is to describe the necessary steps to:

- Set up the Domino server
- Set up Lotus Enterprise Integrator

9.1 Domino server setup

This chapter takes you through the process to install, configure and set up the Domino server on the AS/400 system. The scenario used is shown in Figure 130. DB2, the Domino Web server (HTTP task), and the AS/400 native HTTP server are running on the same AS/400 system, protected by a firewall.

For network protection, a firewall is the most common solution to use. However, in V4R4 of the operating system for AS/400, there is support for IP filters, Network Address Translation, and Virtual Private Network (VPN) among other things, which in some scenarios may prove sufficient to set up the security needed for the AS/400 system to be on the Internet.

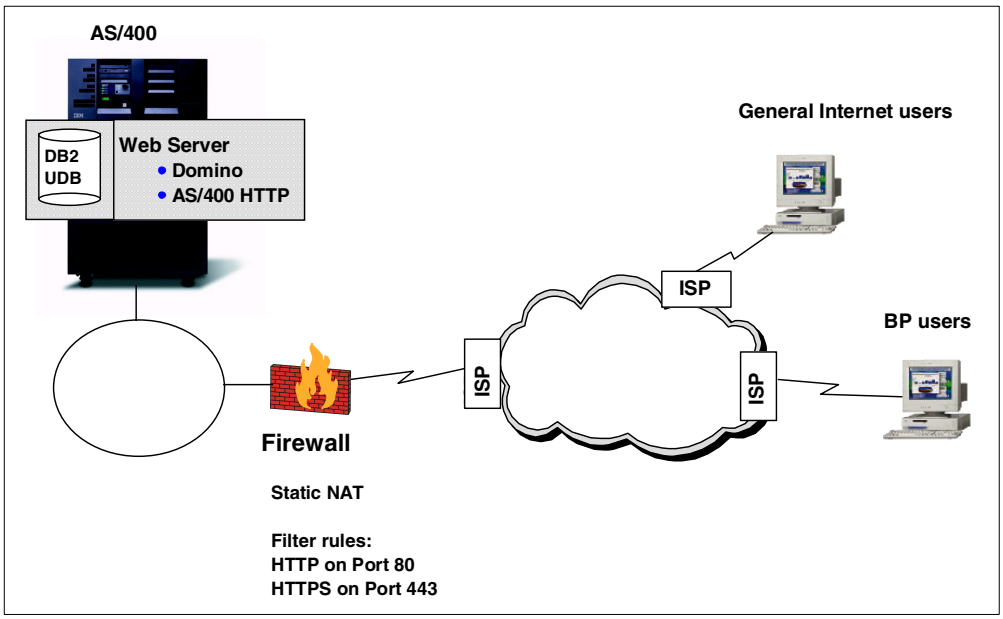


Figure 130. DB2, Domino, and AS/400 HTTP on the same AS/400 system

Another and equally good scenario is where you put your Domino Web server (and in some situations the IBM HTTP server) on a separate AS/400 system, which is placed on the demilitarized zone (DMZ), protected by a firewall. See Figure 131 on page 216 for an example of this.

In this case, you need to access the DB2 on a remote system. To do this, you have to add the necessary entries in the Relational Database Directory. Use the Work with Relational Database Directory Entries (`WRKRDBDIRE`) command. Create an entry defining the remote database, the location, and the type of connection you want to use. See Figure 132 on page 216 and Figure 133 on page 217 for an example.

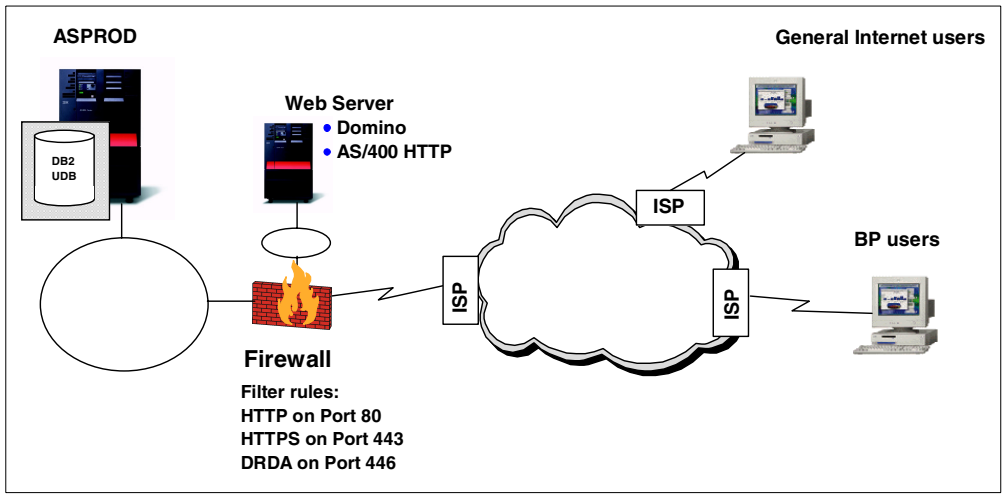


Figure 131. Domino and AS/400 Web servers on a separate system in the DMZ

```
                Work with Relational Database Directory Entries

Position to  . . . . . .

Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display details   6=Print details

         Relational          Remote
Option   Database            Location                Text

         AS15                10.8.6.220              DB2 on remote AS/400
         AS20                *LOCAL                  Local RDB Directory Entry

                                                                        Bottom
F3=Exit   F5=Refresh   F6=Print list   F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 1999.
```

Figure 132. Work with Relational Database Directory Entries command

```
                    Add RDB Directory Entry (ADDRDBDIRE)

Type choices, press Enter.

Relational database  . . . . . > AS25          Character value
Remote location:
  Name or address  . . . . . . . > '10.8.6.28'



Type . . . . . > *IP           *SNA, *IP
Text . . . . . > 'DB2 on remote AS/400'

Port number or service program     *DRDA

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 133. Add RDB Directory Entry command

The Add RDB Directory Entry has to be done on both systems: the Web server system and the DB2 system. They need to know of each other according to the Distributed Relational Database Architecture (DRDA).

As mentioned before, this chapter only covers the implementation shown in Figure 130 on page 215.

9.1.1 Initial configuration

First you need to configure a Domino server on the AS/400 system. The configuration can be done in one of two ways:

- Through the 5250 screens running the Configure Domino Server (CFGDOMSVR) command
- Through the AS/400 Operations Navigator

Both options are described in “Installing and setting up Domino for AS/400” of the redbook *Lotus Domino for AS/400 R5: Implementation*, SG24-5592.

If you work from a 5250 display, you can use the Configure Domino Server (CFGDOMSVR) command. Press the F4 key to access the display shown in Figure 134 on page 218. Table 22 on page 218 shows what you can enter in this display.

```
                    Configure Domino Server (CFGDOMSVR)

Type choices, press Enter.

Server name  . . . . . . . . . . . > DOMSVR5



Option . . . . . . . . . . . . .   *FIRST        *FIRST, *ADD, *REMOVE
Data directory . . . . . . . . . > /lotus/domino/DOMSVR5

Organization . . . . . . . . . . > eBoats

                                                                      More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 134. Configure Domino Server (CFGDOMSVR) (Part 1 of 4)

Table 22. Parameters for CFGDOMSVR (Part 1 of 4)

Parameter	Explanation	The sample value
Server name	Specifies the name of the Domino server. See also the online help for further recommendations.	DOMSVR5
Option	Specifies whether you are setting up the first Domino server, adding an additional Domino server, or removing a Domino server.	*FIRST
Data directory	Specifies the path to the AS/400 directory where you want Domino data files to reside, for example: `/NOTES/DATA` If the specified directory does not exist, it is automatically created with public authority of *EXCLUDE.	/lotus/domino/DOMSVR5
Organization	Specifies the organization name for the Domino server. The organization name can be the name of your company or a group within your company.	eBoats

Press the Page Down key to reach the display shown in Figure 135. Type the values provided in Table 23 for the parameters shown on the screen.

```
                    Configure Domino Server (CFGDOMSVR)

Type choices, press Enter.

Administrator:
  Last name  . . . . . . . . . . > Admin

  First name . . . . . . . . . .
  Middle initial . . . . . . . .                 Character value
  Password . . . . . . . . . . . > welc0me
  Minimum password length  . . . > 7             0-31
  Internet password  . . . . . .   *NONE
Time zone  . . . . . . . . . . . > CST           GMT,EST,CST,MST,PST,CET ...
Daylight savings time  . . . . .   *YES          *YES, *NO
Web browsers . . . . . . . . . . > *ALL          *NONE, *ALL, *HTTP, *IIOP

Internet mail packages . . . . . > *ALL          *NONE,*ALL,*IMAP,*POP3 ...
              + for more values
SMTP services  . . . . . . . . .   *DOMINO       *DOMINO, *MSF

                                                                       More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 135. Configure Domino Server (CFGDOMSVR) (Part 2 of 4)

Table 23. Parameters for CFGDOMSVR (Part 2 of 4)

Parameter	Explanation	The sample value
Administrator	Specifies the password and name of the person who is the administrator for this Domino server. The administrator's name can be your name, another person's name, or a Notes group name. This name will appear in the Server document, in the access control list (ACL) of the Domino Directory (Public Address Book), and in your Notes ID. You can refer to the online help for more recommendations.	Last name: Admin Password: welc0me Minimum Length: 7
Time zone	Specifies the time zone to be used by the Domino server.	Central Standard Time
Daylight savings time	Specifies whether the server time stamp should be adjusted for daylight savings time. This value does not affect the actual system time.	*YES (default)
Web browsers	Specifies which, if any, Web features should be included in the Domino server configuration. These features enable Web browsers to access your Domino server. Choose *HTTP to include the Domino HTTP Web server, *IIOP to include the Internet Inter-ORB Protocol (IIOP) feature, or *ALL to include both.	*HTTP (or *ALL if IIOP is also used)

Parameter	Explanation	The sample value
Internet mail packages	Specifies which, if any, Internet mail packages should be included in the Domino server configuration. These packages enable the Domino server to send mail to and receive mail from the Internet. Choices are: *IMAP, *POP3, *SMTP, *ALL, *NONE.	*SMTP
SMTP service	If you specified *SMTP or *ALL for Internet mail packages (MAIL), use this parameter to specify the type of SMTP support to set up. You can choose between native Domino SMTP, or AS/400 Mail Server Framework including the AS/400 SMTP server.	*DOMINO

Press the Page Down key to access the screen shown in Figure 136. Type in the values provided in Table 24 for the parameters on the screen.

```
                    Configure Domino Server (CFGDOMSVR)

Type choices, press Enter.

Directory services . . . . . .   *LDAP         *NONE, *ALL, *SYSDIR, *LDAP

News readers . . . . . .   *NONE         *NONE, *NNTP
Connection services  . . . . . .   *DECS         *DECS, *NONE
Advanced services  . . . . . > *PARTITION    *NONE, *ALL, *PARTITION...
              + for more values

                          Additional Parameters

Replace configuration  . . . . .   *YES          *YES, *NO
Domain name  . . . . . .   *ORG
Network name . . . . . .   NETWORK1
Country code . . . . . .   *BLANK
Certifier ID . . . . . .   *GEN


                                                                      More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 136. Configure Domino Server (CFGDOMSVR) (Part 3 of 4)

Table 24. Parameters for CFGDOMSVR (Part 3 of 4)

Parameter	Explanation	The sample value
Directory services	Specifies which, if any, directory services should be included in the configuration. *SYSDIR includes directory synchronization services in the configuration. *LDAP includes the LDAP services in the configuration.	*LDAP
News readers	Specifies whether an Internet news reader should be included in the Domino server configuration. A news reader enables the Domino server to send messages to and receive messages from Internet newsgroups. It also enables Internet news readers to access the Domino server.	*NONE (default)

Parameter	Explanation	The sample value
Connection services	Specifies whether the Domino Enterprise Connection Services (DECS) feature should be included in the Domino server configuration. DECS provides the capability to build live links from Domino pages and forms to data from relational databases or ERP systems.	*DECS (default)
Advanced services	Specifies which, if any, features of the Lotus Domino Advanced Services should be included in the Domino server configuration.	*PARTITION
Replace configuration	Specifies whether the existing Domino server configuration files in the directory that is specified in the Data directory (DTADIR) parameter should be erased and replaced with the new ones.	*YES (default)
Domain name	Specifies the domain name for the Domino server. Each Domino domain has a common Domino Directory (Public Address Book) that is shared by all Domino servers in the domain.	*ORG (default) Points to the organization parameter
Network name	Specifies the name of the Notes named network to which this Domino server belongs. A Notes named network identifies a group of servers that share a common protocol so they can communicate directly.	NETWORK1 (default)
Country code	Specifies a two-character country code that is added to the certifier ID for the Domino server. If you are planning to use your Domino server to communicate with other companies, you can use a country code to minimize the chance that another company has the same Domino server ID.	*BLANK (default)
Certifier ID	Specifies the certifier ID file to be used. Each grouping of Domino servers, which is known as a domain, has an organization certifier ID that is stored in a file named CERT.ID. During setup of the first Domino server, Domino automatically creates the organization certifier ID and saves it in the directory that is specified in the Data directory (DTADIR) parameter. This certifier ID automatically certifies the first server's ID and the administrator's user ID. When you register new users or servers, you use the certifier ID to give access to the domain. You also use the organization certifier ID when you create organizational unit certifiers for organizational units (lower level units) within an organization.	*GEN (default)

Press the Page Down key one last time to reach the fourth screen of the CFGDOMSVR command (Figure 137 on page 222). Type in the values provided in Table 25 on page 222 for the parameters on the screen.

```
                    Configure Domino Server (CFGDOMSVR)

Type choices, press Enter.

Administrator ID . . . . . .   *GEN

Server ID  . . . . . .  . . .   *GEN

Start server . . . . . .  . .   *NO           *YES, *NO
Log replication events . . . . .   *YES          *YES, *NO
Log client session events  . . .   *YES          *YES, *NO
TCP/IP port options:
  Encrypt network data . . . . .   *NOENCRYPT    *ENCRYPT, *NOENCRYPT
  Internet address . . . . . . . > '10.8.62.13'
Subsystem and object names . . .   *GEN          Name, *GEN
Collation  . . . . . .  . . .   *STD          *STD,CS,DA-DK-AA,DE,E2-ES ...
Copy Administrator ID file . . .   *ALL          *DOMDIR, *DTADIR, *ALL
Additional services  . . . . . .   *NONE         *NONE,*ICM
              + for more values

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 137. Configure Domino Server (CFGDOMSVR) (Part 4 of 4)

Table 25. Parameters for CFGDOMSVR (Part 4 of 4)

Parameters	Explanation	The sample value
Administrator ID	Specifies the user ID file to be used for the Domino administrator. During setup of the first Domino server, Domino automatically creates a user ID for the Domino administrator and attaches it as a file named USER.ID to the administrator's person document in the Domino Directory (also known as the Public Address Book).	*GEN (default)
Server ID	Specifies the server ID file to be used for the Domino server. During setup of the first Domino server, Domino automatically creates a server ID for the new server. The new server ID file is saved in the directory that is specified in the Data directory (DTADIR) parameter and has the default name of SERVER.ID.	*GEN (default)
Start server	Specifies whether the Domino server should be started when the CFGDOMSVR command completes the setup.	*NO (default)
Log replication events	Specifies whether the replication events should be logged in the Notes log (LOG.NSF) for this Domino server.	*YES (default)
Log client session events	Specifies whether the client session events should be logged in the Notes log (LOG.NSF) for this Domino server.	*YES (default)

Parameters	Explanation	The sample value
TCP/IP port options	Specifies options for the TCP/IP port. Use the Encrypt network data option to specify whether the system should encrypt data that is sent through the port. Encrypting the data can make the transmission more secure, but the transmission speed may decrease slightly. Use the Internet address option to specify a separate Internet (IP) address for the server's port. You might use this option, for example, to specify a separate address for each partitioned server.	Encrypt network data - *NOENCRYPT Internet address: 10.8.62.13
Subsystem and object names	Specifies the name to be used for the AS/400 subsystem in which the Domino server runs. This name is also used for other AS/400 objects that are used for the server. These objects include a job description (*JOBD), a job queue (*JOBQ), a class (*CLS), and possibly BRMS lists.	*GEN (default)
Collation	Specifies the way in which the Domino server should sort characters.	*STD (default)
Copy Administrator ID file	Specifies where the system should copy the Administrator ID file for the server.	*ALL (default)
Additional services	Specifies which, if any, additional services should be included in the server configuration. This is where you specify to use Internet Cluster Manager (ICM).	*NONE (default)

When you finish setting the parameters of the Configure Domino Server (CFGDOMSVR) command, you press the Enter key, and wait until the command is finished. You see the Domino console screen running the configuration as shown in Figure 138 on page 224.

When the message `Press ENTER to end session` appears in the console screen, you have configured the Domino server.

```
  /eBoats successfully registered.
  Creating ID for Admin/eBoats.  This requires 1-3 minutes of computation...
  Begin certifying Admin/eBoats...
  Done certifying Admin/eBoats...
  Begin registering /O=eBoats...
  Adding /O=eBoats to the Name and Address Book...
  /O=eBoats successfully registered.
  Begin registering domsvr5...
  Creating ID for domsvr5/eBoats.  This requires 1-3 minutes of computation...
  Begin certifying domsvr5/eBoats...
  Done certifying domsvr5/eBoats...
  Adding domsvr5/eBoats to the Name and Address Book...
  domsvr5/eBoats successfully registered.
  Begin registering Admin...
  Creating mail file for Admin/eBoats...
  Adding Admin/eBoats to the Name and Address Book...
  Admin/eBoats successfully registered.
  Press ENTER to end terminal session.

===>

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom  F19=Left   F20=Right F21=User Window
```

Figure 138. The Domino console screen

Press the Enter key, and you return to the AS/400 menu from which you came. Here you see the message `Command CFGDOMSVR ended successfully`.

You have now finished configuring your Domino server.

Next, you need to start the Domino server and connect to it from your Lotus Notes client, using the created Administrator ID.

9.1.2 First administration steps

From the Domino Administrator client, you need to register some users, create one or more user groups, and modify some server parameters in the Domino server document.

After you start the Domino Administrator, the Administration window appears. This window has three main areas: the server list, tabs, and tools. The first time you start the Domino Administrator, the system automatically creates a server list.

To administer the Domino server, you select the server from the server list. After you select the server, information about that server appears in all the tabs.

Note

You can also open the server's document in the Domino Directory directory (names.nsf) of the Domino server. To open the server document from a Notes client, click **File->Database->Open**. select the Notes domain address book ("eBOATS's Address book" in our example) on the server.

9.1.2.1 User management

You may need to create additional users. This is an optional step.

We suggest that you create a user group, called Administrators, to have all the users authorized to manage the server in a single group. This will prevent from authorizing every user, who requires it, to any resource they have to manage.

To create a user group, click the **People & Groups** tab. The navigator displayed in Figure 139 appears.

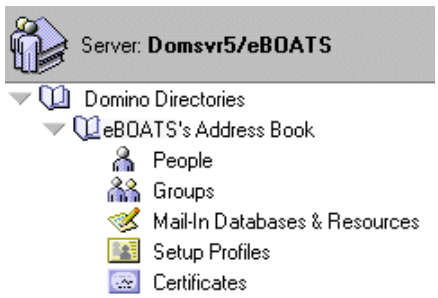


Figure 139. Accessing the Domino directory

If you click People, you can see the user you created during the server setup (Admin in our example). You can create an additional user by clicking Add Person.

If you click Groups, you can create a group called Administrators. Figure 140 shows you the main panel. Here, you add the user Admin. If you created some additional users, they can also be defined as members of the Administrators group.

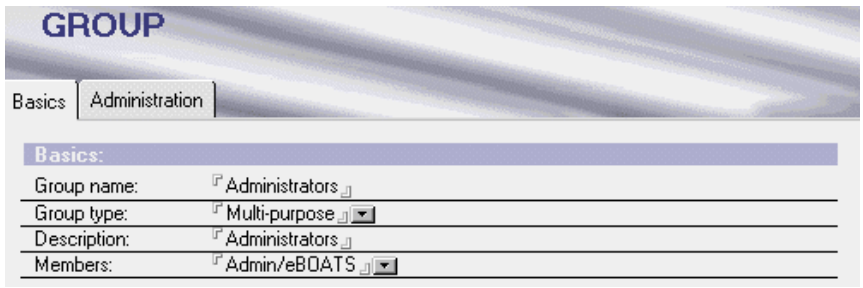


Figure 140. Creating the Administrators group

Figure 141 on page 226 shows you the panel under the Administration tab. The user Admin is the owner and an administrator of the user group. The last parameter on the second panel keeps the default value.

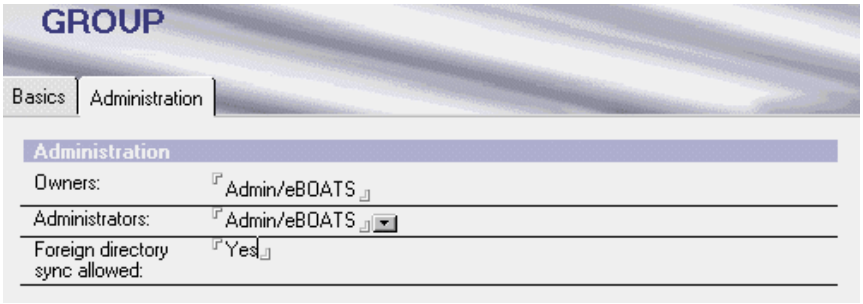


Figure 141. Creating the Administrators group (continued)

Click **Save and Close** to save the user group.

9.1.2.2 Domino server security changes

To help manage the Domino server more smoothly, but also to enhance your current server's security, you now need to authorize the group Administrators in several places in the Domino server document. To access the security information, click the **Configuration** tab, and click the **Edit Server** button. Then, click the **Security** tab.

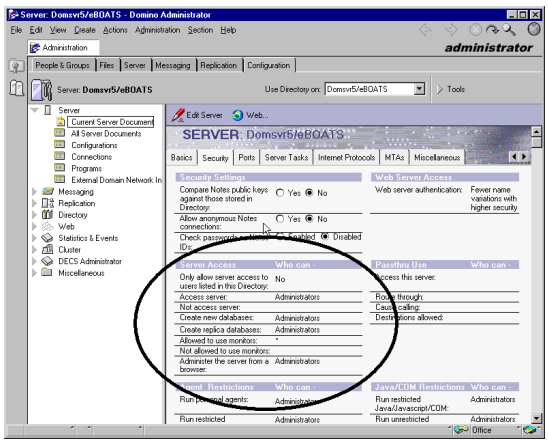
In the Security tab of the server document, set the values as shown in Figure 142. On each parameter, read carefully the message that appears in the information area, at the bottom of the display. For example, for the Create new databases parameter, *none* allows all users to create new databases. However, as soon as one user or user group is defined, nobody else is authorized to create new databases.

Server Access:

Access server	**Administrators** **Business Partners**
Create DB	**Administrators**
Create replica DB	**Administrators**
Admin through browser	**Administrators**

Agent restrictions:

Run personal agents	**Administrators**
Run restricted LotusScript/Java agents	**Administrators**
Run unrestricted LotusScript/Java agents	**Administrators**



Java/COM restrictions:

Run restricted Java/Javascript/COM	**Administrators**
Run unrestricted Java/Javascript/COM	**Administrators**

Figure 142. The security settings in the server document

Notice that the group Administrators is defined in all agent restriction settings. This is done to prevent unauthorized users from running the agents successfully,

and therefore putting in a Trojan horse or getting access to other resources than appropriate.

In this example, the default value for Web Server Authentication is “More name variations with lower security”.

When “More name variations with lower security” is set, Domino tries to authenticate users based on the name and password entered. This authentication method can be vulnerable to hackers who attempt to access a server through a legitimate user account by guessing names and passwords. This choice is the default and is the Web server lookup behavior used in earlier releases of Domino.

“Fewer name variations with higher security” implements a lookup technique that is less vulnerable to attacks, because a single authentication attempt does not produce as many matches. This reduces the likelihood that a guessed password matches. This choice is recommended for tighter security. It requires users to enter only the following information in the name and password dialog box in a Web browser:

- Full hierarchical name
- Common name or Common name with CN= prefix
- Alias name (a name listed in the User name field of the Person document, excluding the first name listed in the field)

9.1.2.3 HTTP settings

To access the HTTP information in the Domino server document, click the **Configuration** tab, and click the **Edit Server** button. Click the **Internet Protocols** tab, and then click the **HTTP** tab in the server document.

First, notice that browsing the Web site is not allowed. This can be seen in Figure 143 on page 228. The Allow HTTP clients to browse databases: parameter, in the HTTP basic settings, is set to No. We recommend you leave it the way it is.

If you test the URL that is used for server browsing (for example `http://domsvr5:8080?OpenServer`), you should get an empty page.

Note

For server browsing to work, databases must reside in the data directory of the Domino server. The URL command to browse the server is:

`http://host?OpenServer`

Server browsing provides users with links to all databases in the data directory. Accesses to each Domino database should be granted through the ACL.

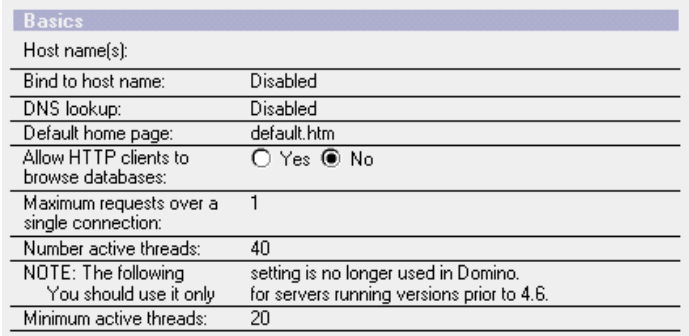


Figure 143. HTTP basic settings

You need to proceed with these changes:

1. Specify the home URL to be `/eboatsinternet.nsf?Open`. This means that if you specify the URL `http://Domsvr5:8080`, you will be directed to the URL `http://Domsvr5:8080/eboatsinternet.nsf`. This way, the users of the eBoats International Web site only have to remember the host name for the URL to access the Web site. The section in the Domino server document, under the HTTP tab, where this setting can be changed, is shown in Figure 144.

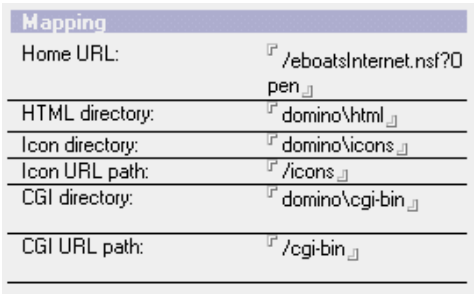


Figure 144. Defining Home URL

2. Allow Web agents to run concurrently.

   The Run Web agents concurrently parameter corresponds to the NOTES.INI DominoAsynchronizeAgents=1 setting which could be defined in Domino R4.6. The Web agent time-out is the maximum number of seconds a Web agent can run.

   The section in the Domino server document, under the HTTP tab, where this setting can be changed, is shown in Figure 145.

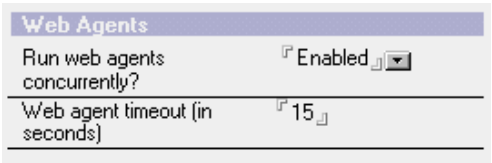


Figure 145. Settings to run Web agents concurrently

3. Allow Java servlet support, as specified in C.3.1, “Setting properties for servlets” on page 359. The section in the Domino server document, under the Domino Web Engine tab, where this setting can be changed, is shown in Figure 146. You select the Domino Servlet Manager.

Java Servlets	
Java servlet support:	Domino Servlet Manager
Servlet URL path:	/servlet
Class path:	domino\servlet
Servlet file extensions:	
Session state tracking:	Enabled
Idle session timeout:	30 minutes
Maximum active sessions:	1000
Session persistence:	Disabled

Figure 146. Settings to run servlets

9.1.2.4 Avoiding conflicts between the AS/400 HTTP server and Domino

The AS/400 operating system (OS/400) includes several TCP/IP application servers, including an HTTP server. This HTTP server processes HTML documents, CGI scripts, and Java scripts for home pages. Domino for AS/400 also provides an HTTP server capability, which enables Notes databases to be seen as HTML documents on the Web.

You can have both HTTP servers installed and running. However, the Domino HTTP server and the AS/400 HTTP server are both set up to use TCP/IP port 80. Because both HTTP servers use the same port, the server that is started second will have a problem accessing the port.

An example of that situation is provided in 9.1.5, “Starting and stopping the Domino Web server” on page 231.

To eliminate this problem, perform one of the following options:

- Change the TCP/IP HTTP server so that it uses a port other than port 80.
- Configure each server to listen on port 80 but at a different IP address.
- Change the Domino HTTP server so that it uses a port number other than port 80. You change the HTTP server port number in the server document of the Domino Directory for the Domino server.

The section in the Domino server document, under the Ports tab and the Internet Ports tab, where this setting can be changed, is shown in Figure 147.

	Web (HTTP/HTTPS)
TCP/IP port number:	8080
TCP/IP port status:	Enabled
Authentication options:	
Name & password:	Yes
Anonymous:	Yes
SSL port number:	443
SSL port status:	Disabled
Authentication options:	
Client certificate:	No
Name & password:	Yes
Anonymous:	Yes

Figure 147. Internet ports

In our sample environment, note the following points:

- The TCP/IP port number is 8080.
- The TCP/IP port status is set to Enabled to allow TCP/IP connections on the port.
- Name & password is set to Yes to use name and password authentication.
- Anonymous is set to Yes to allow anonymous access.

Note

If you also use IBM HTTP Server for AS/400 as a Web server, you need to use the BindSpecific On directive in the server configuration. The On value ensures that the HTTP server binds to the IP address specified in the HostName directive only, instead of binding to all local IP addresses.

See the *HTTP Server for AS/400 Webmaster's Guide*, GC41-5434, for more information on configuring the HTTP server.

The settings for SSL are discussed in 11.2, "Configuring the port for SSL" on page 299.

More information on this topic is available in the Domino for AS/400 Help database. The Domino for AS/400 Help (as400hlp.nsf) database contains all Domino AS/400-specific information. It is also delivered with the software as the printed book *Installing and Managing Domino for AS/400*.

9.1.3 Changes in the NOTES.INI file

One change is necessary in the NOTES.INI file. The JavaUserClasses parameter needs to be added, as specified in C.4.2, "Agent requirement" on page 370.

9.1.4 Setup required to access DB2 UDB for AS/400

As you use several database integration tools, such as LS:DO, @Db functions, or DECS, or LEI, to access the DB2 UDB for AS/400 relational database, you identify the database by providing its relational database name. In our sample application, the relational database name is AS20.

The relational database name is defined by a local relational database entry in the AS/400 relational database directory. You can only have one local relational database entry. Use the Work with Relational Database Directory Entry (`WRKRDBDIRE`) command to determine if a local relational database entry already exists and add an entry if it does not exist.

To determine if a local relational database entry exists on your AS/400 system, follow these steps:

1. On an AS/400 command line, enter:

   ```
   wrkrdbdire
   ```

2. In the Remote Location column, look for an entry named *LOCAL.
   - If a *LOCAL entry exists, look for the name under the Relational Database column on the same line. This name is the relational database name.
   - If a *LOCAL entry does not exist, add one.

To add a local relational database entry, complete these tasks:

1. On an AS/400 command line, enter:

   `wrkrdbdire`

2. Type `1` in the Option field and the name of the relational database in the Relational Database field.
3. Press Enter.
4. Enter `*LOCAL` in the field identified as Remote location: Name or address.
5. Press Enter.
6. Use the default values in the additional fields, and press Enter to add the entry.

Note

The creation of a user profile to access the DB2 UDB for AS/400 collection, tables, and stored procedure is presented in 8.1.2, “Security for the tables and stored procedure” on page 167.

9.1.5 Starting and stopping the Domino Web server

The Domino HTTP task needs to be started so the application can be accessed. This is important since most of the sample application is used through the Web browsers. Table 26 shows the commands that can be used to start and stop the HTTP task.

Table 26. Starting and stopping the Domino HTTP task

To do this	Perform this task
Start the Web server manually.	Enter `load http` at the console.
Start the Web server automatically when you start Domino.	Edit the ServerTasks setting in the NOTES.INI file to include the http command. Domino adds the HTTP task by default to the NOTES.INI file during installation.
Stop the Web server.	Enter `tell http quit` at the console.
Use new server configuration settings without stopping and restarting the HTTP server task.	Enter `tell http restart` at the console. This command results in the HTTP task shutting down, reloading, and refreshing certain Web server settings. It is the equivalent to issuing `tell http quit` followed by `load http`. However, the down time for the server is much shorter because it is not completely reloading all HTTP memory. This command deletes the in-memory page and user-authentication caches.

To verify whether the HTTP task is running, you can enter `show tasks` or `sh ta` at the Domino server console. In the list of tasks, you should find an entry such as:

```
HTTP Web Server      Listening on port(s) 80
```

After starting the server, you may receive such messages as:

```
04/21/2000 10:18:08 AM  JVM: Java Virtual Machine initialized.
04/21/2000 10:18:50 AM  HTTP Socket bind error, hostname/ip S20.eboats.ibm.com
04/21/2000 10:18:50 AM  HTTP server: Could not bind port 80. Port may be in use
04/21/2000 10:18:50 AM  HTTP Web Server shutdown
```

This means the port that the HTTP server is trying to use is already used by another TCP/IP application on your AS/400 system.

To understand which application is using the port or to look for a free port number, you can use the Work with TCP/IP Network Status (NETSTAT) command from a 5250 display.

The easiest way is to enter NETSTAT, and enter option 3 (Work with TCP/IP connection status). Then press F14 (Display port numbers), and press F13 (Sort by column). Finally, select **Local port.**

You can scroll through the display until you can see the port number you are looking for, as shown in Figure 148. Option 5 (Display details), allows you to see information about the application using the port.

```
                    Work with TCP/IP Connection Status
                                                                System:   AS20
Local internet address  . . . . . . . . . . . :   *ALL

Type options, press Enter.
  4=End   5=Display details

     Remote           Remote  Local
Opt  Address           Port    Port  Idle Time  State
     *                    *     53  217:13:09  *UDP
     *                    *     53  217:13:09  *UDP
     *                    *     67  000:03:21  *UDP
     *                    *     80  000:52:26  Listen
     *                    *    137  217:13:06  Listen
     *                    *    137  000:00:44  *UDP
     *                    *    138  000:00:43  *UDP
     *                    *    139  000:32:32  Listen
     10.8.61.199       1025    139  001:36:01  Established
     10.8.61.152       1293    139  000:32:32  Established
     *                    *    161  043:09:48  *UDP
                                                                        More...
F5=Refresh   F11=Display byte counts   F13=Sort by column
F14=Display port names   F15=Subset by local address   F24=More keys
```

Figure 148. Using the NETSTAT command

9.1.6 Other useful HTTP console commands

There are some additional commands that can be useful to manage the HTTP task. They are presented in Table 27.

Table 27. Other HTTP commands

To do this	Perform this task
To show connected users	Enter `tell http show users` at the console. **Note**: This command can be used only if the server is configured to use session-based authentication (in the Domino Web engine settings, as described in 10.3, “Enabling session-based authentication” on page 252) for the Web. This command shows the User Name, IP address, and the time of expiration, which is 30 minutes by default. The results display only users who are authenticated; anonymous users cannot be tracked.
To display information about file system protection on the machine and each virtual server	Enter `tell http show file access` at the console.
To display the current status on the use of SSL for the server, and each virtual server or virtual host	Enter `tell http show security` at the console.
To display a list of all configured virtual servers or virtual hosts running on the server	Enter `tell http show virtual servers` at the console.

9.2 LEI server configuration

You can configure your LEI server by using the Add LEI Server (ADDLEISVR) command. To do this, perform the following steps:

1. Start the Domino server that is to run LEI. Ensure that the Domino server is started by using the Work with Domino Servers (`WRKDOMSVR`) command. Make sure that the user profile you are signed in with has the *ALLOBJ, *SECADM, *IOSYSCFG, and *JOBCTL special authorities.
2. Since the group Administrators has been authorized specifically to create databases on the Domino server (refer to 9.1.2.2, “Domino server security changes” on page 226, for more details), you need to change the server document again and authorize the Domino server itself, or the LocalDomainServers group (as shown in Figure 149 on page 234), to create databases.

Server Access	Who can -
Only allow server access to users listed in this Directory:	No
Access server:	
Not access server:	
Create new databases:	Administrators, LocalDomainServers
Create replica databases:	Administrators
Allowed to use monitors:	*
Not allowed to use monitors:	
Administer the server from a browser:	Admin/eBoats, Administrators

Figure 149. Authorization to create Domino databases given to the local servers group

3. Enter `ADDLEISVR` on a command line, and press Enter.
4. Complete the parameters presented on the first display (see Figure 150), and press Enter.

```
                        Add LEI Server (ADDLEISVR)


LEI server . . . . . > LEISVR5
Local Domino server  . . . . . > DOMSVR5

Domino server for databases  . . > DOMSVR5

Create admin database  . . . . .   *YES          *YES, *NO
Autostart LEI  . . . . . .  .  .   *YES          *YES, *NO
Setup realtime support . . . . . > *YES          *YES, *NO
Migrate from NotesPump . . . . .   *NO           *YES, *NO
Alternate database path  . . . .   *DEFAULT

Admin database . . . . . .  .  .   LEIADM.NSF
```

Figure 150. Adding LEI Server (Part 1 of 2)

When you press Enter, the second display (Figure 151) prompts you for additional configuration parameters.

```
                        Add LEI Server (ADDLEISVR)

Type choices, press Enter.

Database manager:
  Last name  . . . . . .  .  .   Admin

  First name . . . . . .  .  .
  Middle initial . . . . . .  .                  Character value
Log database . . . . . .  .  .   LEILOG.NSF

Install documentation  . . . . .   *Yes          *YES, *NO
Script vault database  . . . . .   LEIVLT
```

Figure 151. Adding LEI Server (Part 2 of 2)

The previous example creates an LEI server called LEISVR5 on a Domino Server called DOMSVR5. The LEI Administrator databases are placed on the same Domino Server (DOMSVR5). The LEI databases are created and the manager is Admin.

Notice that the database manager must match an authorized Notes user identification exactly as the user name appears in the Domino Directory.

The installation modifies the NOTES.INI file.

These are examples of the lines that should appear after configuring your LEI server:

```
EXTMGR_ADDINS=decsext,LIBLEIEXT
ServerTasks=DECS,DIIOP,LDAP,HTTP,Replica,Router,Update,Stats,AMgr,Adminp,Sched
,CalConn,Event,QNNINADD,dircat,LEI
```

In the first line, *LIBLEIEXT* corresponds to the real-time support setup. In the second line, *LEI* appears since the Autostart LEI parameter is set to `*Yes`.

After the LEI server is added to the Domino Server, you are ready to start the LEI server. To start the LEI server from the Domino console, type `LOAD LEI`, and press Enter.

After starting LEI manually (or stopping and starting the Domino server again to start the LEI server automatically), you can access the Domino LEI Administrator database (Figure 152) from a Notes client. It is understood that you should use the Admin user identification, because it is the only user authorized to access the database.

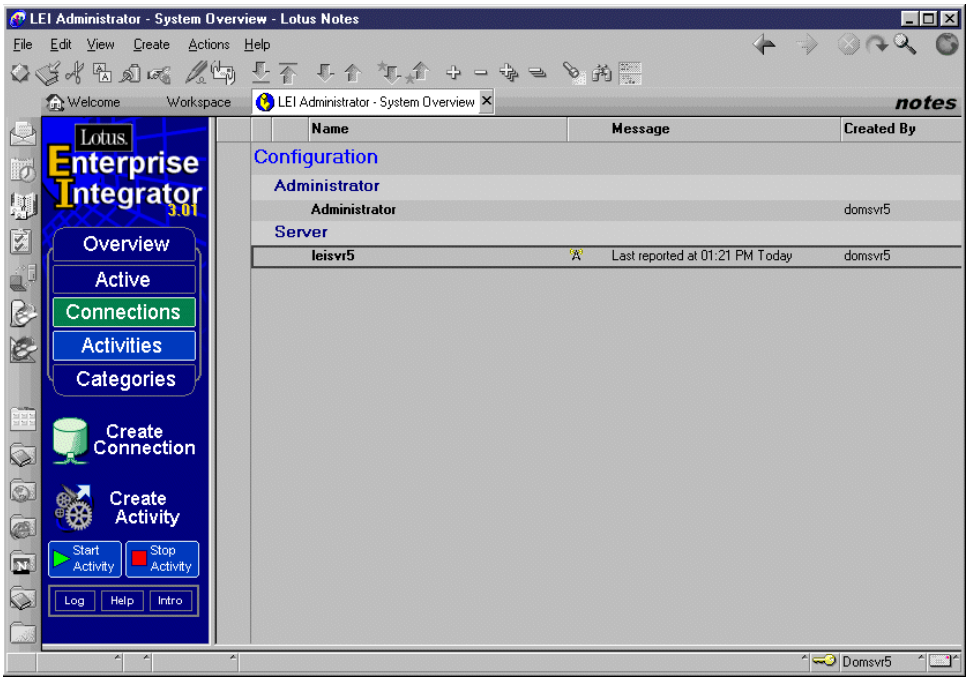


Figure 152. LEI Administrator database after LEI setup

If you look at the LEI Administrator configuration, you find that the status broadcast interval is set to 60 minutes.

If you look at the LEI Server configuration, the current status should be active. You also find that:

- The poll interval is set to 60 seconds. This is the interval at which the LEI Server polls the Administrator database to see if there are any activities that it needs to execute.
- The maximum number of activities is set to 0. This is the greatest number of concurrent activities that the server will be allowed to run. Once the server is running the maximum number of activities, it postpones additional activities until existing activities are concluded. Zero stands for no maximum.
- The maximum duration of activities is set to 0 minutes. This is the longest duration that the server will run any single activity. The server automatically closes any activity that exceeds this duration. Zero stands for no time-out.
- The maximum consecutive failures parameter is set to 5.
- The activity shutdown request time-out is set to 120 seconds.

For more information about using Lotus Enterprise Integrator with Domino on the AS/400 system, you can refer to the “Lotus Enterprise Integrator/Lotus NotesPump” chapter in the redbook *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345.

Note

After you successfully complete the steps required in this chapter, you may decide to save the environment before going further in the security steps.

For recommendations about saving the Domino server and the LEI server, refer to 11.6, “Saving your Domino server configuration and environment” on page 303.

Chapter 10. Sample project user management

This chapter covers three main topics:

- Registering users
- Setting up directory assistance
- Managing the authentication process

10.1 Managing users

eBoats International's Domino Web site supports access for four major user groups:

- Administrators
- Employees
- Business partners
- Anonymous users

10.1.1 Administrators

The administrators have the authority to manage the databases of the eBoats International application. From the Notes client on the internal network, they can access the databases for administration and maintenance.

If they are accessing the databases from the Internet, the ACL for the database in the advanced settings limits the authority given through the setting `Maximum Internet name and password`. All the administrators are members of the user group Administrators (see Figure 153). Administrators are registered in the eBoats address book (names.nsf).

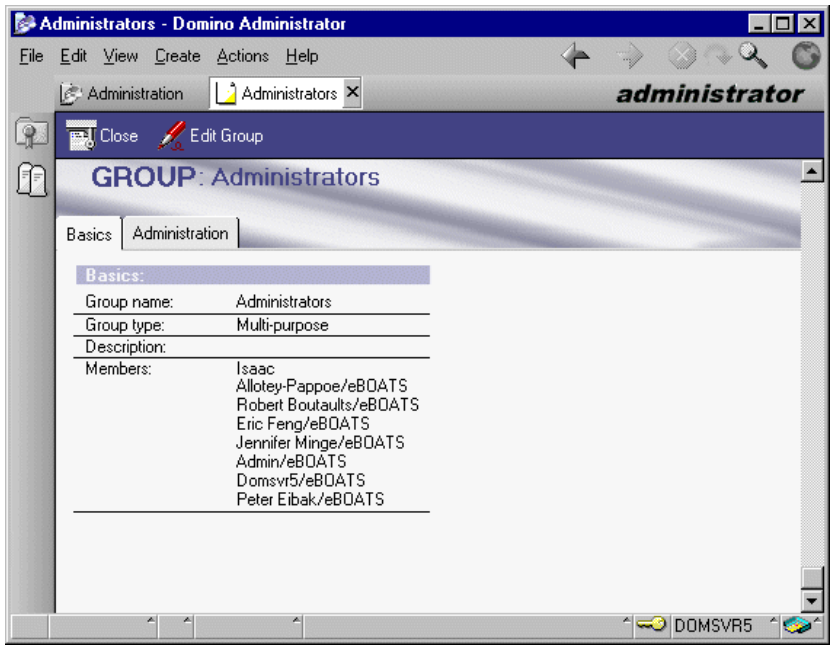


Figure 153. The Administrators user group

10.1.2 Employees

The employees of eBoats International are given appropriate access to the databases where necessary, but have no access to the business partner database eboatsB2B.nsf.

The employees are grouped in the user group Employees, as shown in Figure 154. Employees are registered in the eBoats address book (names.nsf).

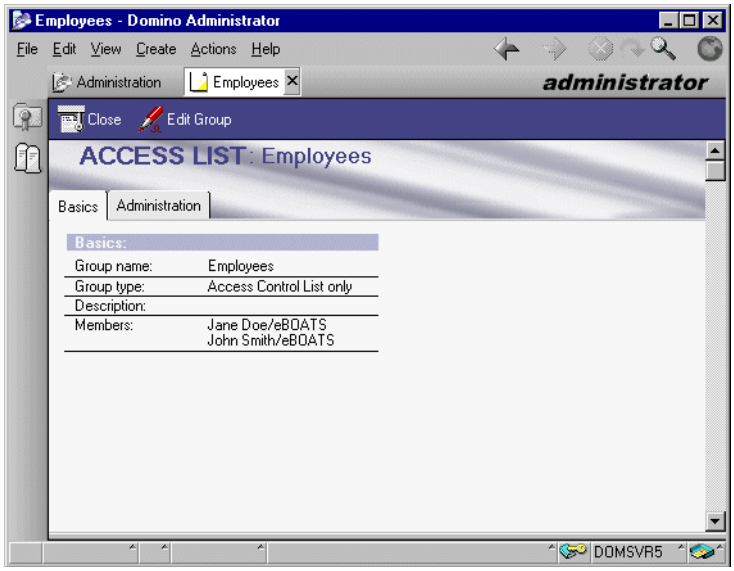


Figure 154. Employees user group

10.1.3 Business partners

The business partners have access to the necessary databases with the appropriate authority, but have no access to the intranet database eboatsIntranet.nsf.

The business partners are registered in the eBoats Web Address Book, eboatsWebNAB.nsf, and are grouped in the user group BusinessPartners registered in the eBoats Address Book names.nsf.

Both registration and additions to the BusinessPartners user group happen after those two events, which are described in 12.1.4, “BP Application” on page 311:

- The user applies to become a business partner.
- The authorized eBoats International employee has the responsibility to approve the application.

Figure 155 shows the registered business partners in the eBoats Web Address Book.

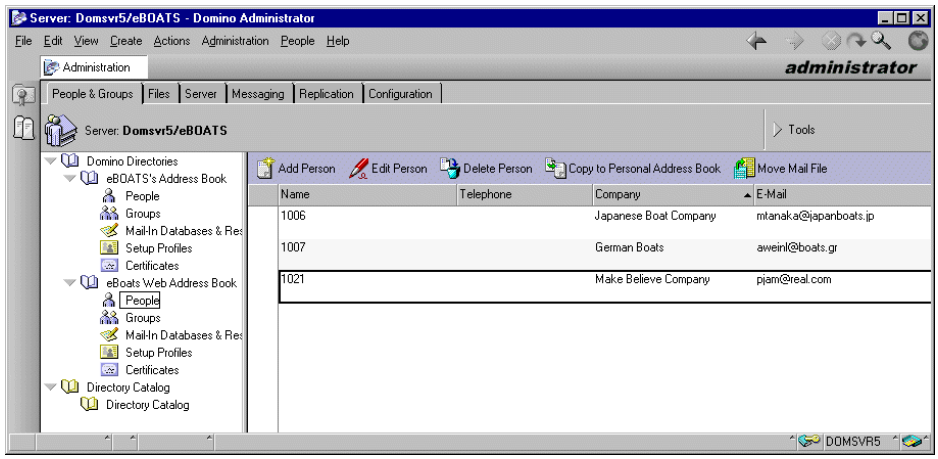


Figure 155. The registered business partners

Figure 156 shows the BusinessPartner user group that is registered in the eBoats Address Book NAMES.NSF.

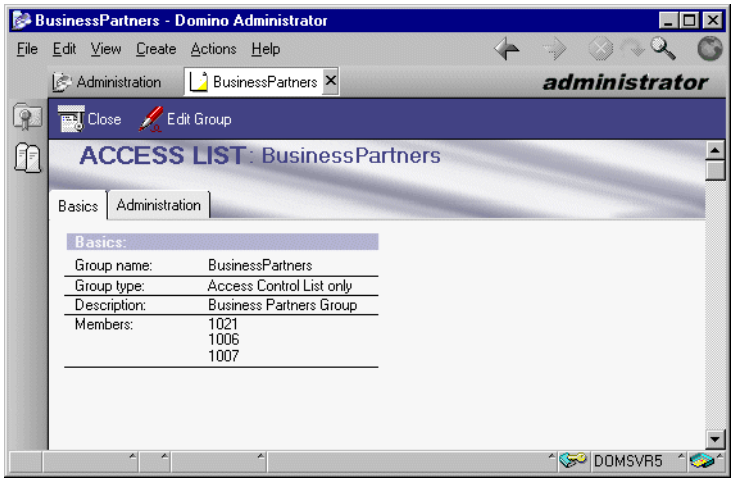


Figure 156. The BusinessPartners user group

10.1.3.1 eBoats Web Address Book creation

The directory that contains the business partner addresses needs to be created. We created the eBoats Web Address Book database, eBoatsWebNAB.nsf, from the pubnames.ntf template.

To create the database, perform the following steps:

1. From the Workspace, select **File->Database->New...**
2. Enter the Name of the server, where the directory would be located, in the server field.
3. Click the **Template Server...** button, and select or enter the name for the templates server.
4. Select the **Show advanced templates box**.
5. Select **Domino Directory** from the template files.
6. Type in a title in the Title field.

7. Type in a name for the Domino database.
8. Click **OK**.

The settings for the eBoats Web Address Book database are shown in Figure 157.

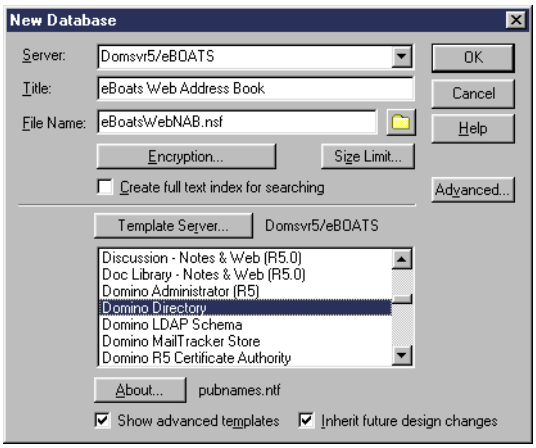


Figure 157. The eBoats Web Address Book database setting

10.1.4 Anonymous Internet users

EBoats International is supporting access for anonymous users in the Internet database, eboatsinternet.nsf, but they have no access to the business partner database, eboatsB2B.nsf, or the intranet database, eboatsIntranet.nsf.

These anonymous users are not registered in any Address Book, so they are not required to authenticate themselves. Access is provided through the ACLs of the databases if Anonymous is added to the ACL and given the appropriate authority.

An example is shown in Figure 158.

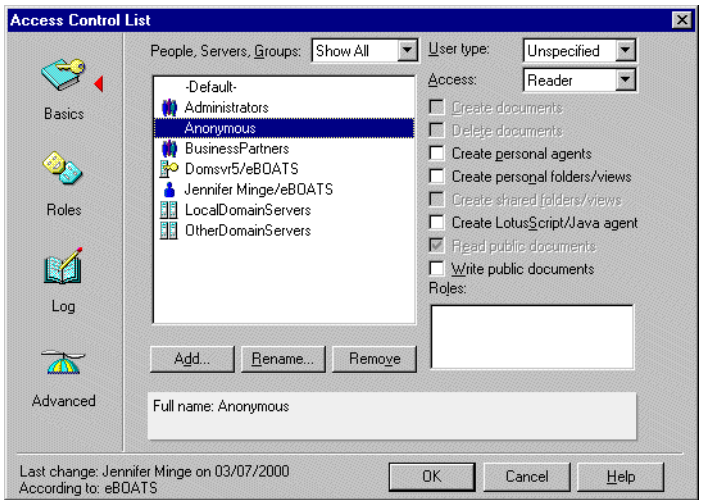


Figure 158. Anonymous access in the ACL

10.2 Setting up directory assistance

In this section, we perform the following steps:

- Create a Directory Assistance Domino database.
- Create a Directory Catalog Domino database.
- Perform changes in the Domino server document.

10.2.1 Creating the Directory Assistance database

We create the Directory Assistance Domino database, da.nsf, from the da50.ntf template.

To create the database, complete the following steps:

1. From the Workspace, select **File->Database->New...**
2. Enter the Name of the server where the directory would be located, in the server field.
3. Click the **Template Server...** button, and select or enter the name for the template server.
4. Select **Directory Assistance** from the template files.
5. Type in a title in the Title field.
6. Type in a name for the Domino database.
7. Click **OK**.

The settings for the eBoats Directory Assistance database are shown in Figure 159.

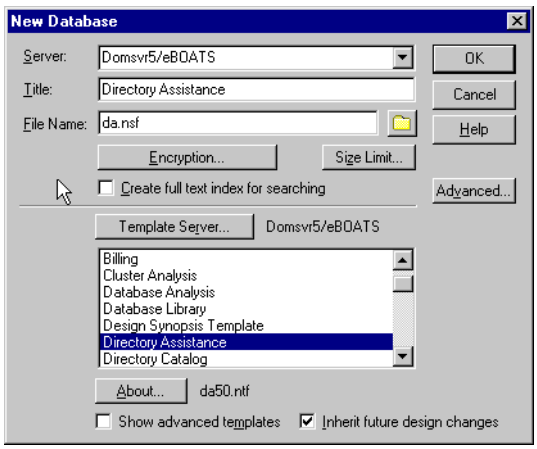


Figure 159. The eBoats Directory Assistance database setting

10.2.2 Creating the Directory Catalog database

We create the Directory Catalog Domino database, dircat.nsf, from the dircat5.ntf template.

To create the database, perform the following steps:

1. From the Workspace, select **File->Database->New...**
2. Type a title in the Title field.

3. Type a name for the Domino database.
4. Enter the Name of the server where the directory would be located in the server field.
5. Click the **Template Server...** button, and select or enter the name for the template server.
6. Select **Directory Catalog** from the template files.
7. Click **OK**.

The settings for the eBoats Directory Catalog database are shown in Figure 160.

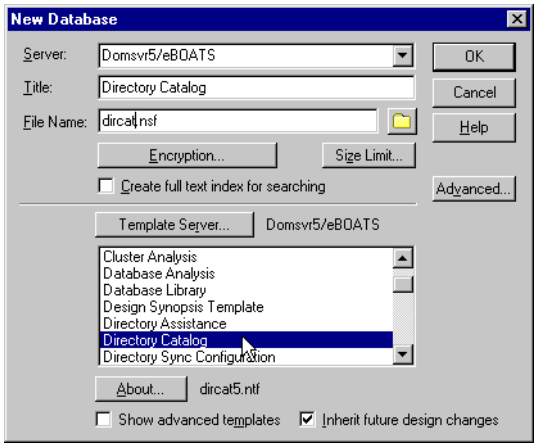


Figure 160. The eBoats Directory Catalog database setting

Once the Directory Catalog database is created, we need to create the Directory Catalog configuration.

To create the configuration, perform the following steps:

1. Open the database you just created, and select **Create - Configuration**.
2. Complete the Directories to include field. Enter the file names of the secondary directories to include in the directory catalog. The only secondary directory we use here is: eBoatsWebNAB.nsf.
3. In the Basics tab, edit the Additional fields to include field to customize the fields included in the catalog. Do not remove the default fields because they are used for optimized quick mail addressing. To look up Web client passwords in the directory catalog when the Web server authenticates the clients, add the HTTPPassword field.
4. In the Basics tab, change the settings for the Sort by field to Last name. We expect users to enter last names followed by first names. This is especially true for business partners which get a number stored in the Last Name field.

The configuration document is shown in Figure 161.

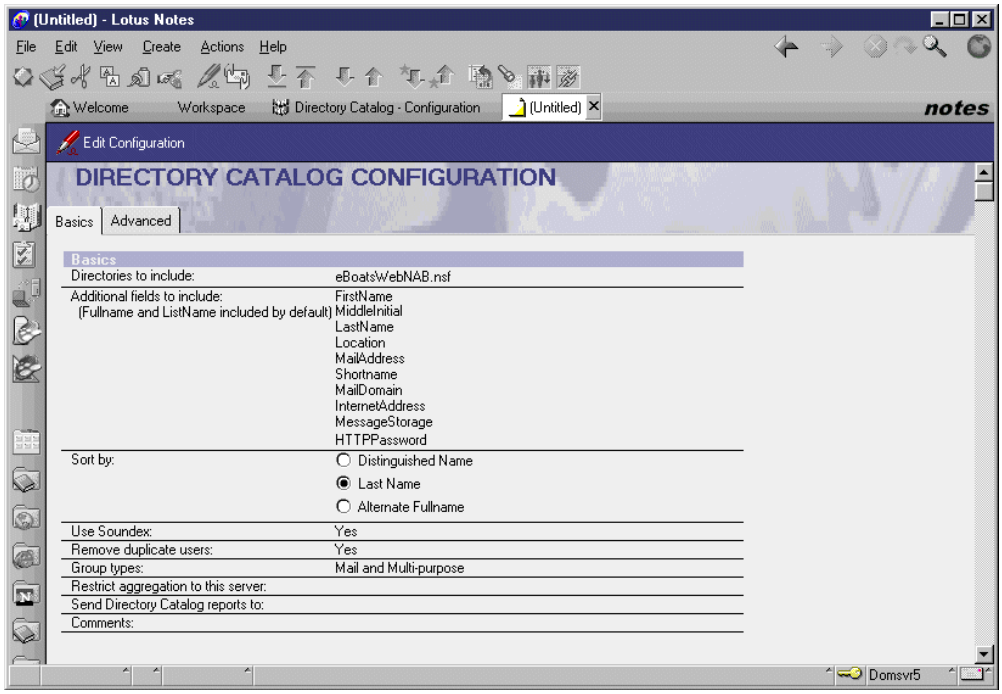


Figure 161. Directory Catalog Configuration

10.2.3 Making changes in the Domino server document

We also have to perform some changes in the Domino server document. Follow these steps:

1. Go the server document (Figure 162).

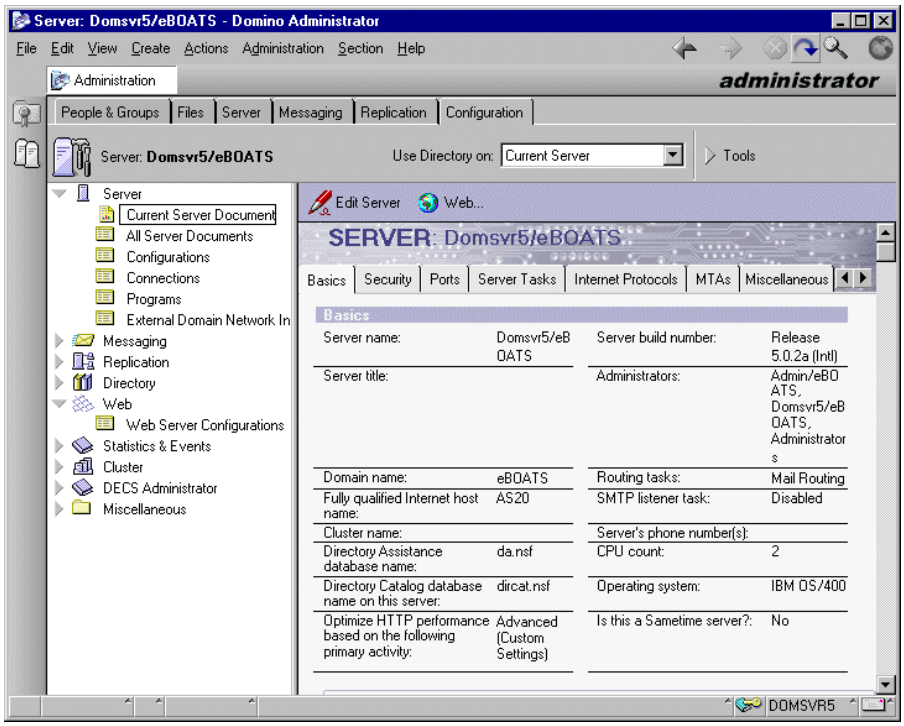


Figure 162. The server document Basics tab

2. Set the values of Directory Assistance and the Directory Catalog as shown in Table 28.

Table 28. The Directory Assistance and Directory Catalog databases

Label	Value
Directory Assistance database name	da.nsf
Directory Catalog database name on this server	dircat.nsf

10.2.4 Adding directory assistance entries

To add directory assistance entries, complete these steps:

1. From the Domino Administrator, in the server pane on the left, select the server that stores the Directory Assistance database. If you don't see the server pane, click the servers icon.
2. Click the **Configuration** tab.
3. Expand **Directory**, as shown in Figure 163.
4. Click **Directory Assistance**.

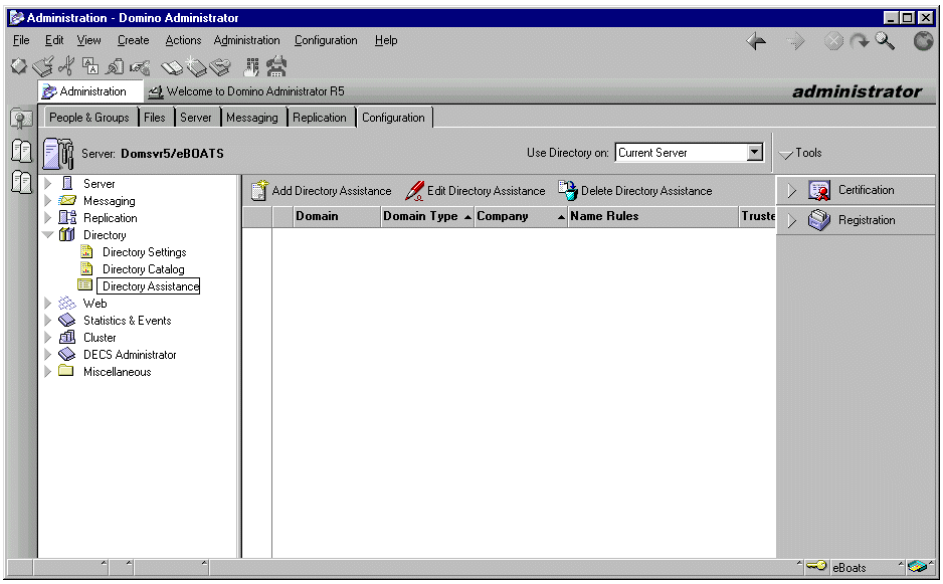


Figure 163. Server directory settings

5. Click **Add Directory Assistance**. You need to complete this step twice, to take into account first the Domino Directory itself (the eBoats Address Book) and an additional directory created to record the business partners' names, passwords, and addresses.
   a. First you create a directory assistance entry for the Domino directory itself. In the Basics tab (Figure 164), complete these fields as described in Table 29. Enter `1` for the search order, and enable it.

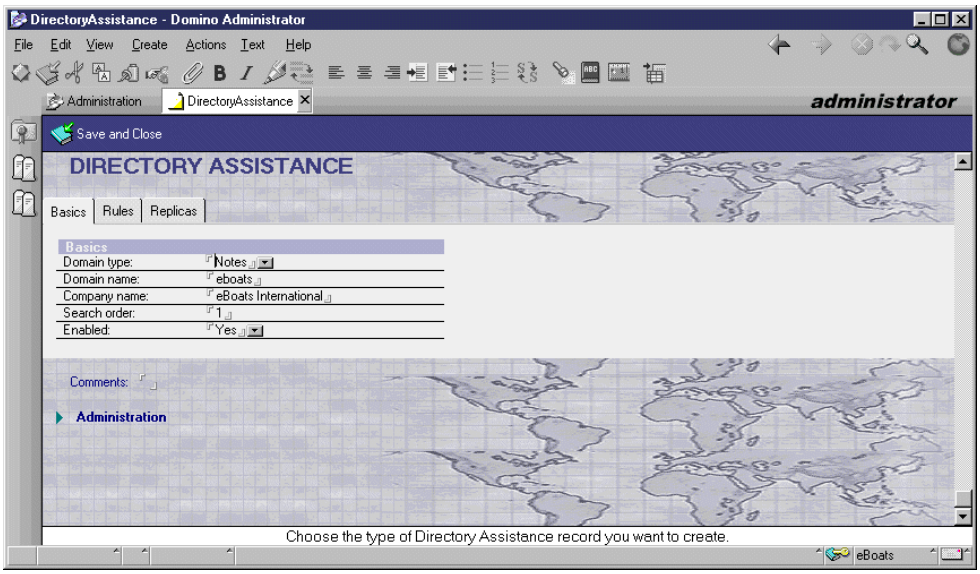


Figure 164. Directory Assistance setup (Page 1 of 3)

Table 29. Directory Assistance Basics fields

Field	Enter
Domain Type	Choose Notes.
Domain Name	The name of the Notes domain associated with the secondary directory. The domain name must be different from the primary Notes domain and from all other domain names configured in directory assistance. If the secondary directory is not associated with a Notes domain, invent a domain name; do not specify the primary Notes domain.
Company Name	The name of the company associated with this directory. Multiple directory assistance documents can use the same company name.
Search Order	A number representing the order in which this directory is searched, relative to other directories in the Directory Assistance database.
Enabled	Choose Yes to enable directory assistance for this directory.

b. Click the **Rules** tab (Figure 165 on page 246), and then complete the fields, as described in Table 30 on page 246.

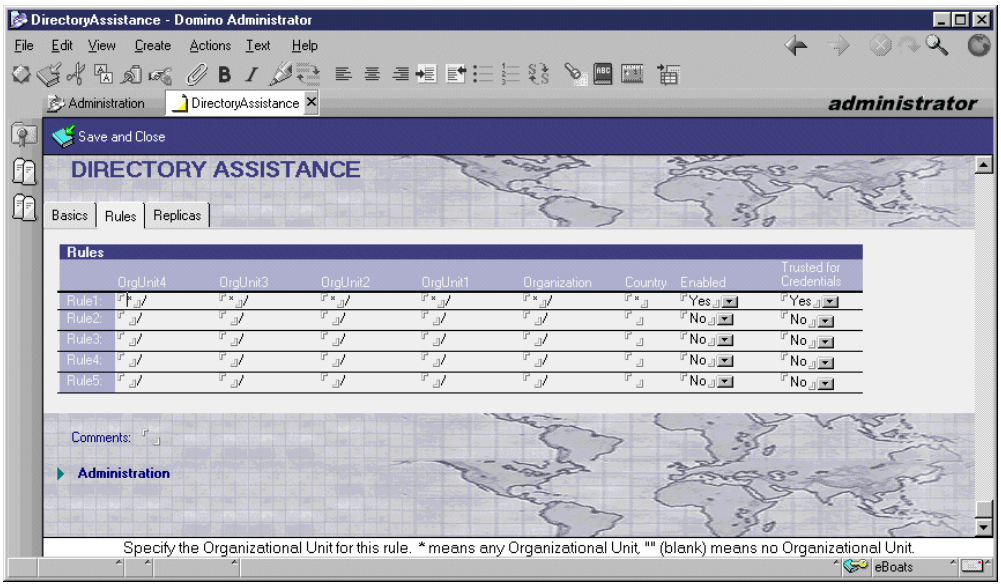


Figure 165. Directory Assistance setup (Page 2 of 3)

Table 30. Directory Assistance Rules fields

Field	Enter
Rule #	One or more rules that describe the names in the directory. By default, the first rule contains all asterisks, indicating all names in the directory.
Enabled	Choose one: - No to disable a specific rule - Yes to enable a specific rule By default, the first rule is enabled.
Trusted for Credentials	Choose one: - Yes to allow Domino to use this directory to authenticate Web clients with names that correspond to the rule - No (default) to prevent Domino from using this directory to authenticate Web clients

c. Click the **Replicas** tab (Figure 166), and complete the fields provided in Table 31 to specify up to five replicas of the secondary directory to use for directory assistance. You use a link to the NAMES.NSF database.

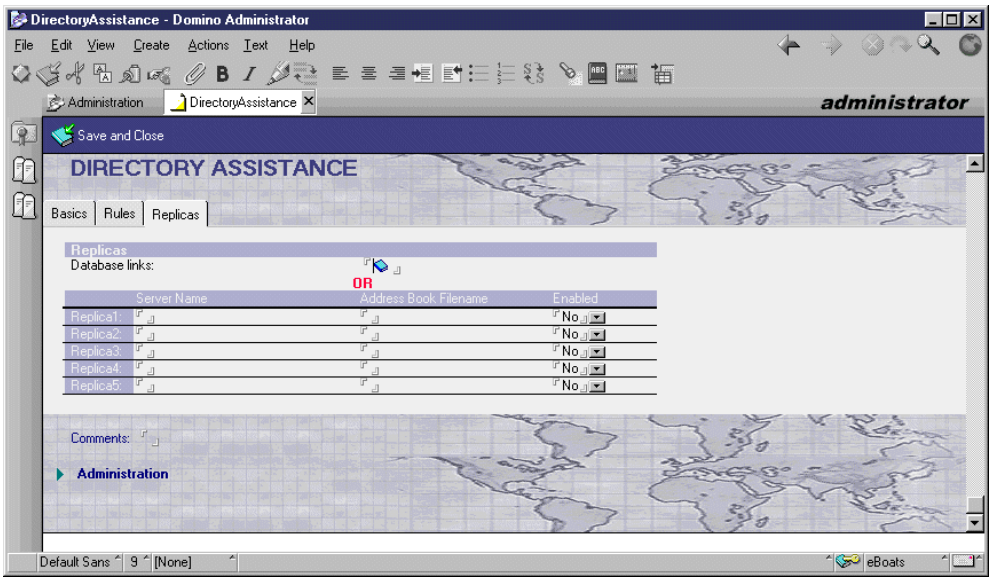


Figure 166. Directory Assistance setup (Page 3 of 3)

Table 31. Directory Assistance Replicas fields

Field	Enter
Database links	Open the replica of the secondary directory, and then choose **Edit->Copy As Link->Database Link**. Select the "Database links" field, and then choose **Edit->Paste**. Use database links only on Domino Release 5 servers. Using database links may delay server startup because when you restart a server that uses directory assistance, server tasks need to retrieve database information from the remote servers to which the links refer. Use database links only if the servers the links refer to are consistently available.
Replica #	The server name and file name of each replica of the secondary directory, for example: Server name: Mail/West/eBoats Directory file name: EBoatsWebNames.nsf Selected Enabled next to each replica.

d. Click **Save and Close**.

The Business Partners are registered in the eBoats Web Address Book, eboatsWebNAB.nsf. This database has been specially created for the project, using the same template as for the Domino Directory. Figure 167 on page 248 shows the registered business partners in the eBoats Web Address Book.

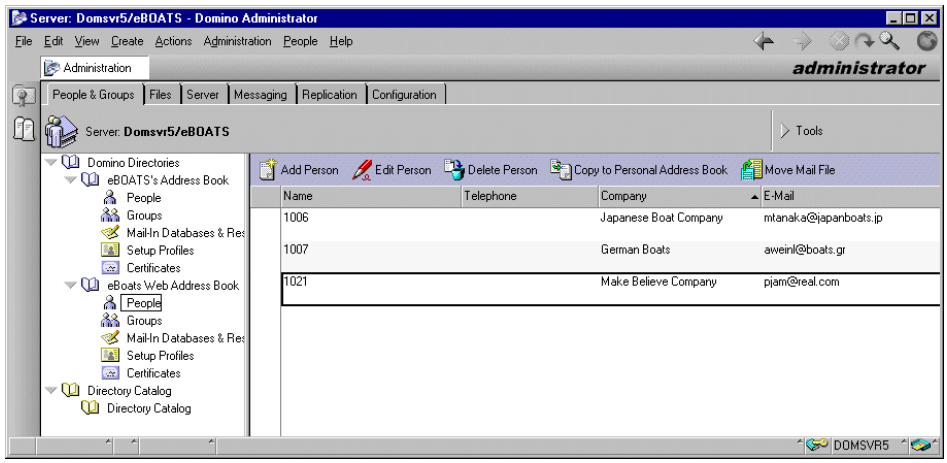


Figure 167. The registered business partners

They are also grouped in the BusinessPartners user group registered in the eBoats Address Book, NAMES.NSF, as explained in 10.1.3, “Business partners” on page 238.

You create a second directory assistance entry for the Domino directory. In the **Basics** tab, complete these fields as described in Table 29 on page 245 (see Figure 164 on page 245). Enter 2 for the search order, and enable it.

Click the **Replicas** tab, and complete these fields to specify up to five replicas of the secondary directory to use for directory assistance. You use a link to the database eboatsWebNAB.nsf.

Note

If you authenticate Web clients registered in the secondary directory and you also use the directory catalog, be sure to include in the Replicas tab the replica of the secondary Domino directory you used to build the source directory catalog. If you specify a replica in the Database links field and in the Replica field, the Replica field takes precedence.

After configuring the source directory catalog, you run the Directory Cataloger task, also known as the Dircat task, on the server that stores the source directory catalog. The Dircat task summarizes entries for users, groups, mail-in databases, and scheduling resources from all the configured Domino directories and combines the entries into the source directory catalog.

If you build a mobile directory catalog, you use a User Setup Profile to replicate it to Notes clients. If you build a server directory catalog, you replicate it to Domino servers that you want to use it. To use a server directory catalog, you must include the catalog file name in the Public Directory Profile or in Server documents.

In general, you should build a source directory catalog on one server in your organization, and then replicate it to servers in other Notes domains, rather than building a source directory catalog in each domain.

Run the Dircat task regularly to keep the source directory catalog synchronized with directories combined in it. Also set up schedules to regularly replicate the source directory catalog to the clients and servers throughout the organization that use it.

The Dircat task can be run on schedule or manually. Running it on schedule is the recommended way.

Perform the following steps to run the Dircat task on schedule:

1. From the Domino Administrator, select the server in the server pane on the left that stores the source directory catalog. If you don't see the server pane, click the **servers** icon.
2. Click the **Configuration** tab.
3. Expand **Server** and select **Current Server Document**.
4. Click **Edit Server** in the Server document.
5. Click the **Server Tasks - Directory Cataloger** tab.
6. Complete the fields listed in Table 32, and then click **Save and Close**.

Table 32. Dircat field content

Field	Enter
Directory Catalog filenames	The file names of the source directory catalogs. Separate multiple file names with commas.
Schedule	Select Enabled.
Run Directory Catalog aggregator at	A time range or one or more specific times to update the source directory catalog. Separate multiple time entries with commas (,).The default is the range 08:00 AM to 10:00 PM.
Repeat interval of	A number representing the minutes between updates that are scheduled during a time range. The default is 360 minutes (every six hours).
Days of week	The days of week to update the directory catalog. The default is daily.

Figure 168 on page 250 shows an example of a schedule. The repeat interval and the days of the week depend on how quickly information for a new user needs to be replicated (how quickly a new business partner is expected to access the eBoats International Web site once their registration is confirmed).

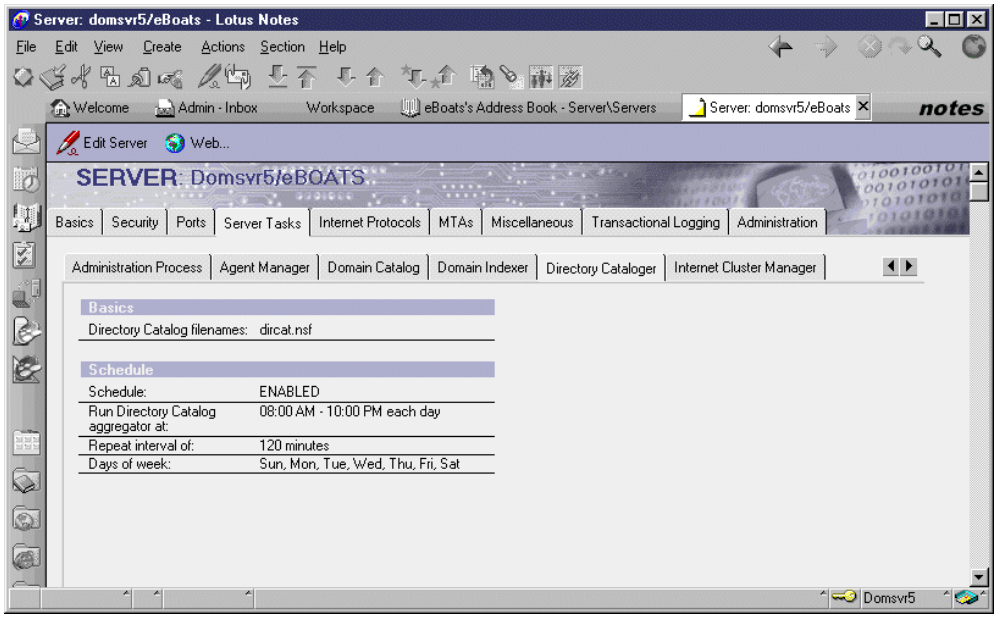


Figure 168. Directory Cataloger schedule

Use these steps to start the Dircat task manually:

Note

When you manually start the Dircat task if the task is already running, Domino displays a message indicating the task is running. If the Dircat task is currently running (for example, if it's already running according to schedule), you must stop the task before you can start it manually. You can't run multiple Directory Cataloger tasks at the same time.

1. From the Domino Administrator, in the server pane on the left, select the server that stores the source directory catalog. If you don't see the server pane, click the **servers** icon.
2. Click the **Server - Status** tab.
3. Click **Console** at the top of the window, enter the command `tell Dircat quit` in the command line box at the bottom. Press Enter.
4. Select the command line box again and enter the load `Dircat dircat.nsf` command (where dircat.nsf is the file name of the source directory catalog). Press Enter.

10.2.5 LDAP service setup

This part is optional. It may be considered if you need to start up the Domino LDAP service.

To use the LDAP service on Domino, you need to perform these tasks:

1. Create a full-text index for the replica of the Domino Directory on the server that runs the LDAP service. We strongly recommend that you create a full text index unless LDAP users search only for names.
2. Start the Domino server, and then start the LDAP task.

3. To allow clients to connect to the LDAP service over the Internet, connect the server that runs the LDAP service to an Internet service provider (ISP) and register the server's DNS name and IP address with the ISP.
4. If your organization uses more than one Global Domain document, you must specify the one that the LDAP service uses to return users' Internet addresses to LDAP clients. Open the Global Domain document. In the Use as default Global Domain field, choose **Yes**.
5. Set up LDAP clients to connect to the LDAP service.
6. (Optional) Customize the default LDAP service configuration. In most cases, the LDAP service functions correctly when using the default settings.
7. To check whether you set up the LDAP service correctly, use an LDAP client or `lsearch` (which is a utility for querying LDAP servers) to issue a query to the LDAP service. A graphical example of an LDAP accessing the eBoats Domino LDAP server is shown in Figure 169.

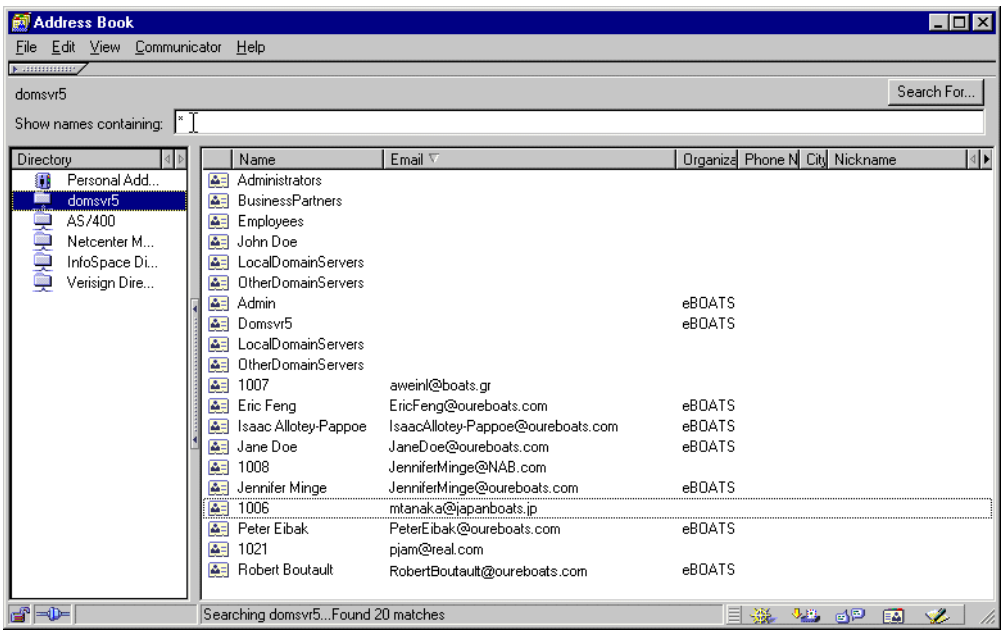


Figure 169. Domino LDAP search from an LDAP client

You can set up directory assistance for secondary Domino directories to:

- Use security credentials in the directories to authenticate Web clients that use the Domino Web server.
- Extend LDAP client searches to the directories.
- Allow Notes users to easily address mail to users registered in the directories.
- Set up directory assistance for LDAP directories to:
  - Use security credentials in the LDAP directories to authenticate Web clients that use the Domino Web server.
  - Verify membership in groups registered in an LDAP directory.

– Refer LDAP clients that connect to a Domino LDAP service to the directories.

– Allow Notes users to verify mail addresses of users in the LDAP directories.

The following steps are necessary to create a Directory Assistance document for a secondary Domino Directory.

Before you create a directory assistance document, make sure that you already:

- Set up the Directory Assistance database
- Planned locations for replicas of the secondary Domino Directory
- Enabled access to the locations chosen for replicas of the secondary Domino Directory

Note: The second and third tasks are not needed in this environment.

If you use a server directory catalog, optionally set up the directory catalog to include the secondary Domino Directory. If you use name and password authentication to authenticate Web clients registered in the directory and you want to store the password in the directory catalog, add the HTTPPassword field to the source directory catalog configuration. You can also store x.509 certificates in the directory catalog, but this significantly increases the size of the directory catalog.

If you want to extend LDAP searches from the Domino LDAP service's primary Domino Directory to this directory, create a full text index on the replica of the directory used by directory assistance. Also optionally perform any of the following tasks to customize the LDAP configuration for this directory:

- Customize which fields anonymous LDAP users can access.
- Optimize searches for the replicas of the directory used by directory assistance.
- Allow LDAP users to add, modify, and delete entries in the directory.
- Enable LDAP searches in alternate languages.

Note

If you allow LDAP clients to modify entries in a secondary Domino Directory, don't use a directory catalog on the Domino server that runs the LDAP service.

10.3 Enabling session-based authentication

To enable session-based authentication, you need to edit the Domino server document. You must also create a Person document for each Web client who will use session-based name-and-password authentication. Note that for Business Partners, the creation of a person document in the eBoats Web Address Book, eboatsWebNAB.nsf, and the creation of an entry in the group Business Partners, in the Domino directory, are performed by the `RegisterBP` agent, as explained in 10.1.3, “Business partners” on page 238.

The section in the Domino server document, under the Domino Web Engine tab, where this setting can be changed, is shown in Figure 170.

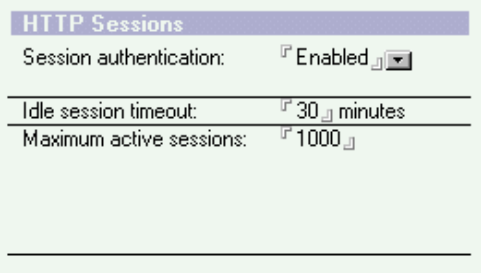


Figure 170. Enabling session-based authentication

You can specify a default log-out time period to log the Web client off the server after a specified period of inactivity. Automatically logging a user off the server prevents others from using the Web client to impersonate a user if a Web client leaves the workstation before logging off. Users can also append *?logout* at the end of a URL to log off a session (for example: `http://Domsvr5/db.nsf?logout`).

You can specify the maximum number of user sessions allowed on the server at the same time. If the server performance is slow, you can reduce the number of active, concurrent user sessions.

10.4 Authentication process

Let us look at what we want to accomplish for the authentication process that takes place when a user on the Internet wants to access the eBoats International Web site. Figure 171 shows how the authentication process for Internet users is being done. The focal point for authentication is the database ACLs.

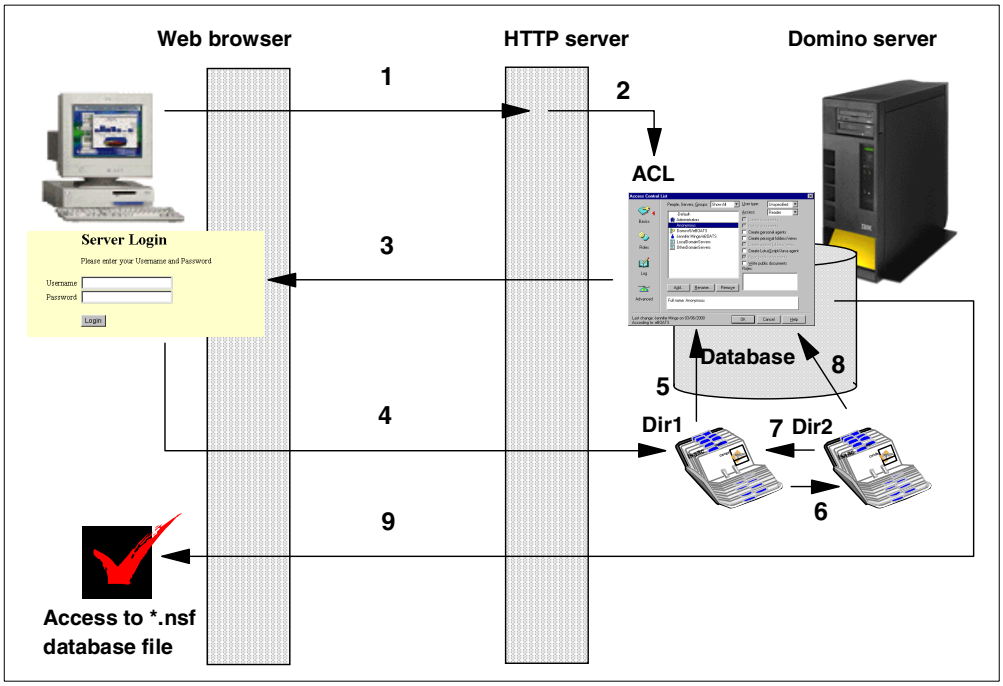


Figure 171. The authentication process for Internet users

The process shown in Figure 171 is explained here:

1. The Web browser is requesting the URL for the Domino Web server.
2. The ACL for the database is being checked for allowing anonymous access. If yes, access to the database will be given through step 9. This is the case for the eboatsinternet.nsf database.
3. If anonymous access is not allowed, a sign on prompt is sent to the browser for authentication. This is the case for the four other databases: eboatsB2B.nsf, eboatsIntranet.nsf, eboatsbp.nsf, and eboatscustomer.nsf.
4. The user enters their user ID and password and returns it to the Domino Web server.
5. The user ID and password are being checked in the Domino directory service for existence. If the user ID and password are in directory 1, it is being checked in the database ACL for access. If it is in the ACL by user or user group, access to the database is given through step 9.
6. If the user ID and password do not exist in directory 1, it is being checked for existence in directory 2.
7. If the user is not in the database ACL, directory 1 is checked for membership of a group.
8. The user or user group is checked for existence in the database ACL.
9. If the user or user group is in the database ACL and access is allowed, access to the database is given.

For this authentication process to work, you need to know your users and user groups even if they are anonymous. When you know your users and the access they need to the applications and databases, you can set up the ACLs for the databases to accomplish the authentication process.

10.4.1 Managing accesses

The application of eBoats International consists of five databases, as described in 8.2, “Domino part” on page 171. Figure 172 shows the eBoats International databases that have to be secured.

Attention

In this section, we detail only the security settings for the application databases. However, you need to review the Access Control List (ACL) settings for each .NSF file on your Domino server, including the user mail databases.

In addition to checking the ACL for your application databases, be sure to check the ACL of system databases such as NAMES.NSF, LOG.NSF, ADMIN4.NSF, DOMCFG.NSF, and so on.

What is the “-default-” access control set to? What is “Anonymous” set to? What is the Maximum Internet name and password access set to in the Advanced ACL settings?

There are also two servlets that need to be protected. The steps to protect the servlets are described in 10.4.1.1, “Protecting servlets” on page 258.

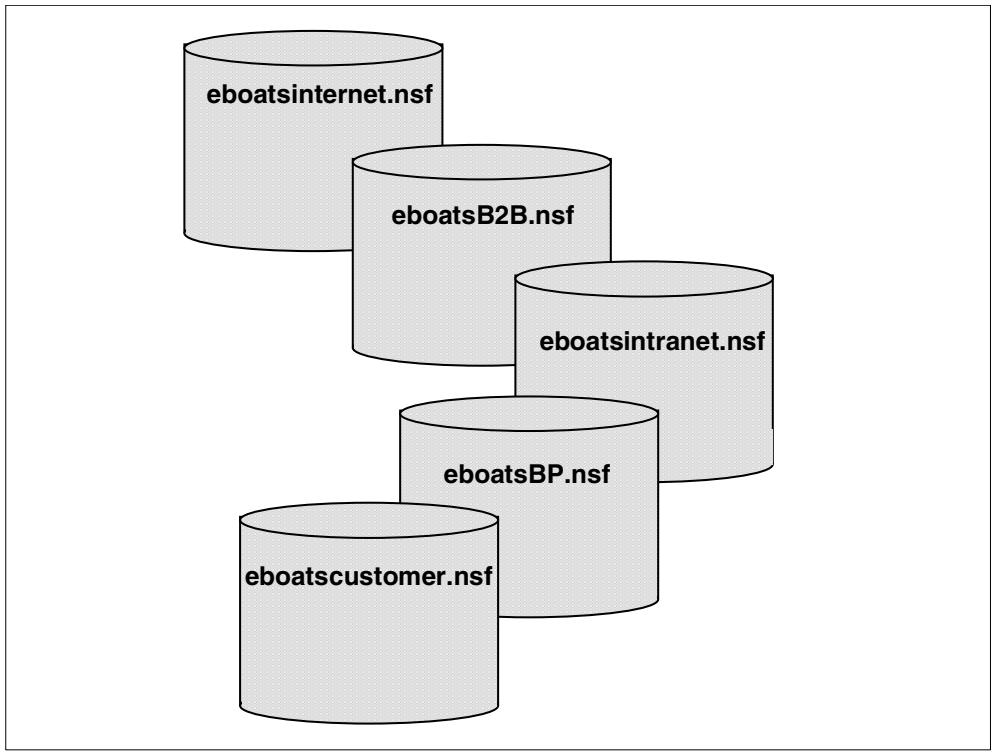


Figure 172. The sample databases

For every other databases on the Domino server, we recommend that you set the Default access in the ACLs to `No Access` to make sure that no anonymous access can be made to these databases from the Internet. You will need explicit access in the Authorization Control Lists (ACLs) for your user or user group to access these databases.

To accomplish the security infrastructure that we need, we implemented the ACLs for the five application databases as shown in Figure 173.

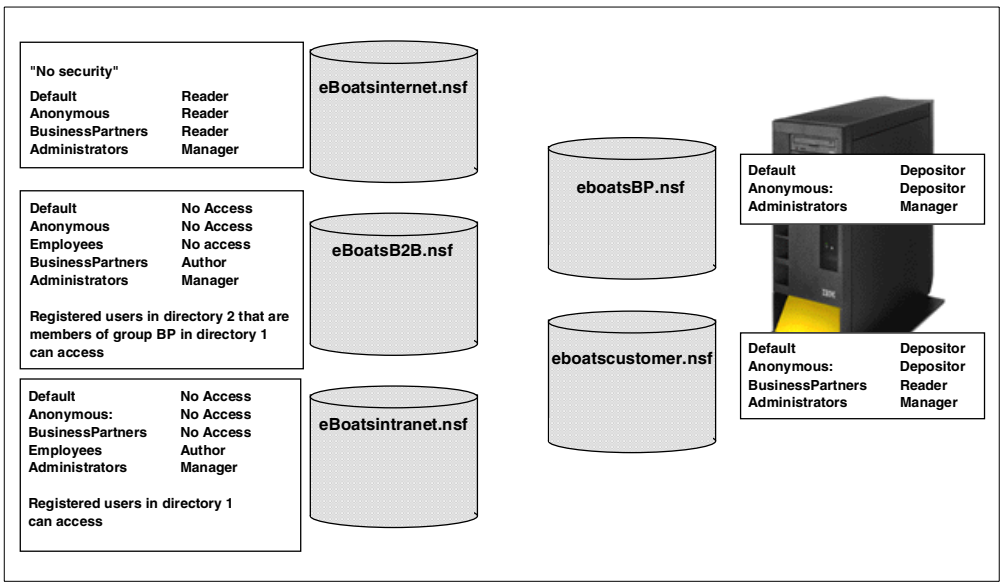


Figure 173. The main elements of the database ACLs

eboatsinternet.nsf
The open Internet database has the authorities shown in Figure 174.

Figure 174. The ACL for eboatsinternet.nsf

It is necessary that everyone has reader access as a minimum.

eboatsB2B.nsf
The business partner database has the authorities shown in Figure 175.

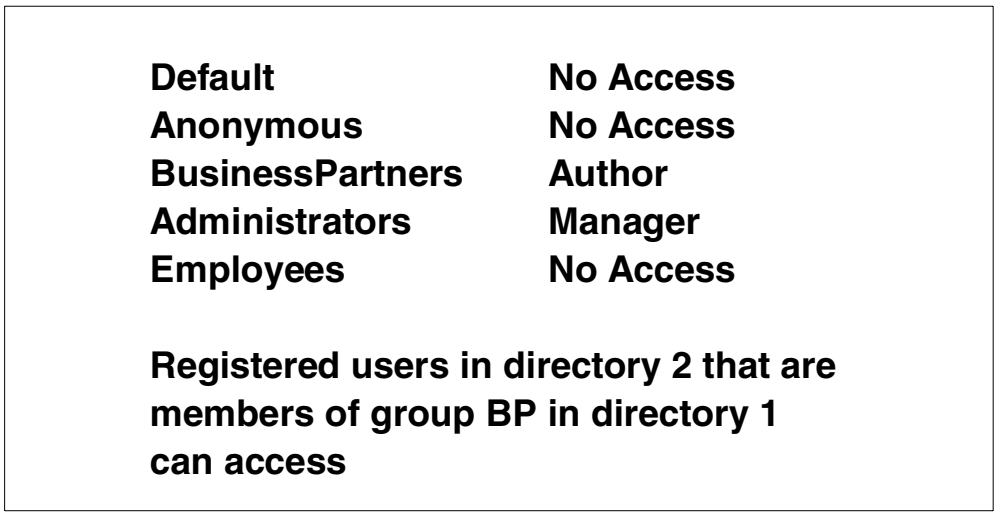


Figure 175. The ACL for eboatsB2B.nsf

Anonymous is given No Access because we want to force users being authenticated through prompting for a user ID and password, which means that you have to be a registered user in one of the of eBoats International Address Book. Business partners have Author rights to create and read their own documents.

eboatsIntranet.nsf
The intranet database has the authorities shown in Figure 176.

Figure 176. The ACL for eboatsIntranet.nsf

The database can only be accessed by employees or administrators at eBoats International. Employees have Author rights to create and read their own documents. External users, such as business partners and Anonymous have No Access.

eboatsbp.nsf
This database has the authorities shown in Figure 177.

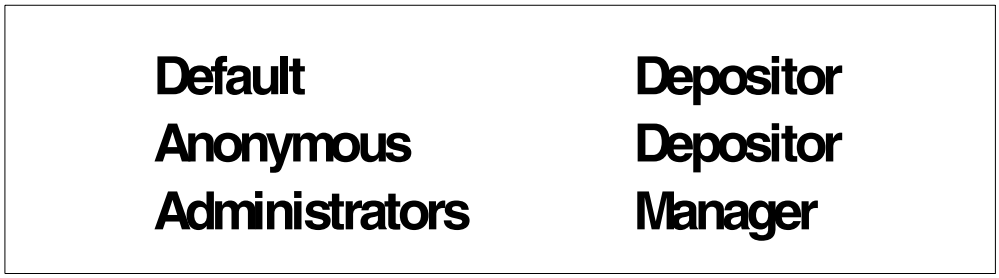


Figure 177. The ACL for eboatsbp.nsf

Anonymous users need to have Depositor rights so they can create documents in the eboatsinternet.nsf database.

eboatscustomer.nsf
The customer database has the authorities shown in Figure 178.

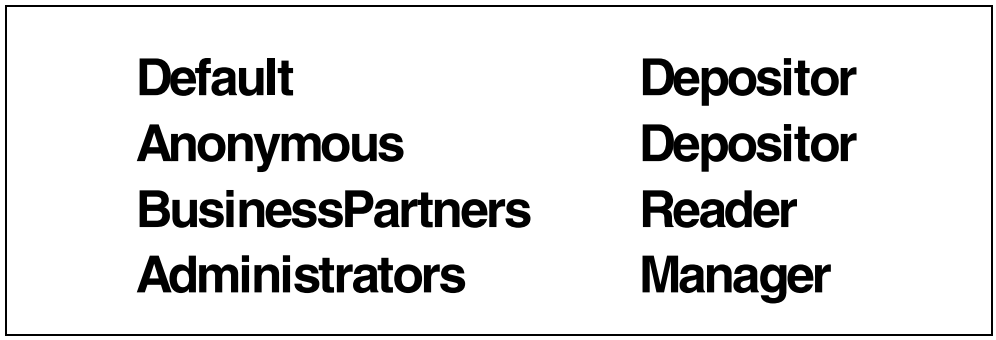


Figure 178. The ACL for eboatscustomer.nsf

Anonymous users need to have Depositor rights to create documents in the eboatscustomer.nsf database.

Business partners need Reader rights to access customer information in the eboatscustomer.nsf database.

Authorities when accessing the databases from the Internet
Using ACL for the databases is the basis for protecting the databases from unauthorized access. However, we want to strengthen the security when the databases are accessed from the Internet.

The eBoats International application is using the parameter Maximum Internet name & password under the advanced option in the ACL. For the five databases, we set the value as shown in Figure 179.

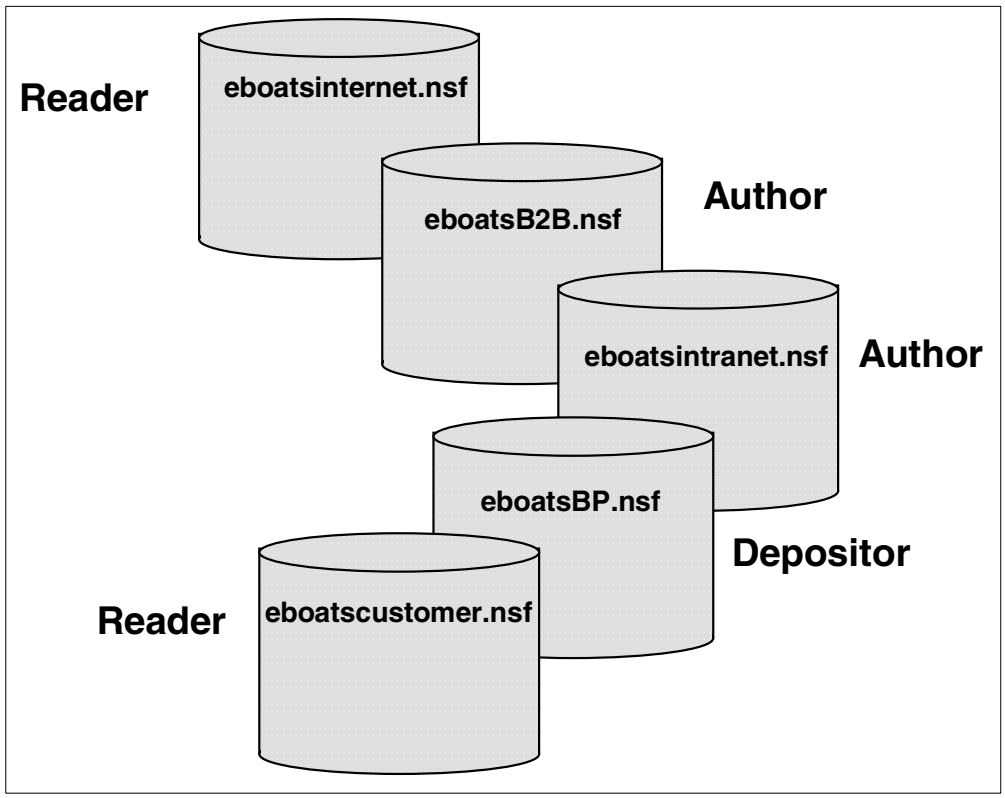


Figure 179. The maximum Internet name and password

This setting is of great value if using user IDs from the Internet with high authorization, such as administrator profiles. You can use them but they are being limited in the authorization they can use when connecting from the Internet.

In that way, you are safer from being hacked successfully from the Internet with a user ID with high authority on the databases.

10.4.1.1 Protecting servlets

To protect files on a server's hard drive, you can create a file protection document for each servlet or for the directory containing the servlet files.

File protection documents control the access that Web browser clients have to the files. You can force file system security for files that browser users can access

(for example, HTML, JPEG, and GIF) and specify the level of access for these types of files and the names of the users who can access them.

You can apply file system protection on CGI scripts, servlets, and agents. However, file protection does not extend to other files accessed by the scripts, servlets, or agents. For example, you can apply file protection on a CGI script that restricts access to a group named “Web Admins”. However, if the CGI script executes and opens other files (or causes other scripts to be executed), the file protection document is not checked to determine whether “Web Admins” has access to these files.

However, file system protection applies to files that access other files, for example, HTML files that open image files. If a user has access to the HTML file but does not have access to the JPEG file that the HTML file uses, Domino does not display the JPEG file when the user opens the HTML file.

You can create a file protection document for a directory or for an individual file. The default path is the Domino data directory. You can also create file protection documents for other directories.

It is recommended setting up file protection documents for all directories accessible to Web users.

To set up file protection for the directory containing the servlets, you have to set up mapping and redirect settings in the Domino server document:

1. Start Domino Administrator.
2. Open the Domino server document as shown in Figure 180.

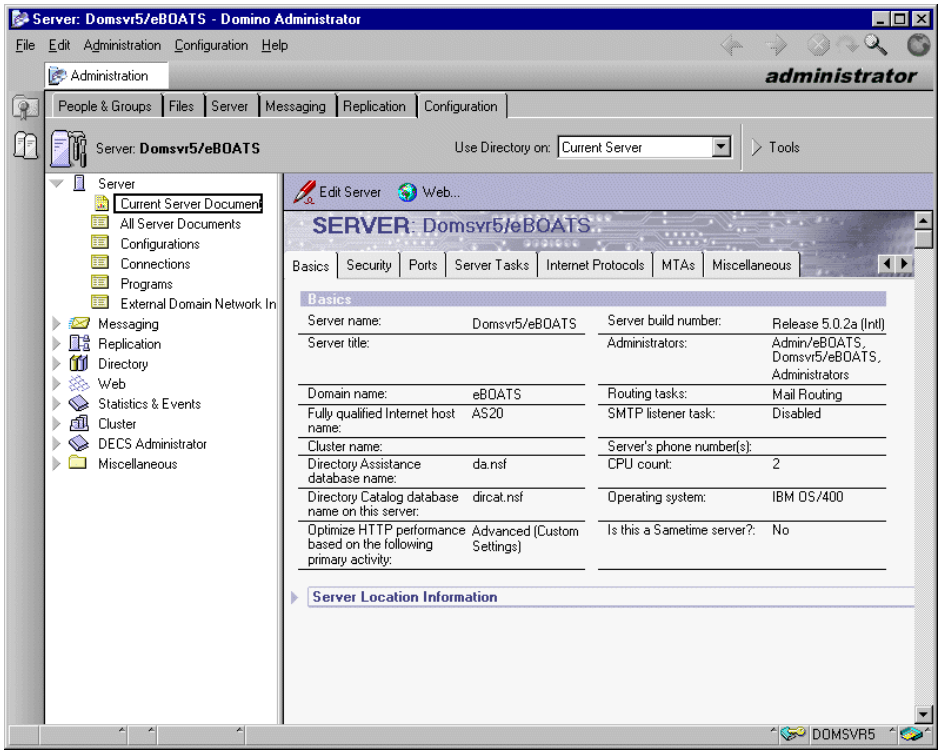


Figure 180. Domino server document

3. From the server document, click the **Web** option as shown in Figure 181.

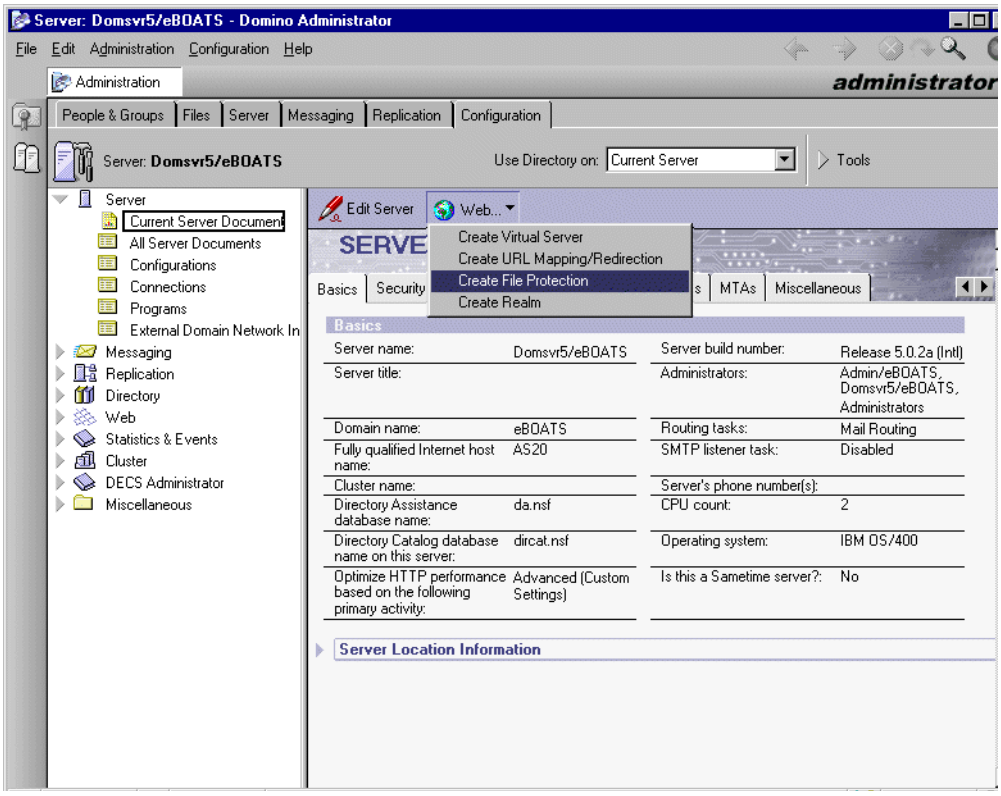


Figure 181. The Web pull-down menu

4. Click the **Create File Protection** menu option. The display shown in Figure 182 appears.

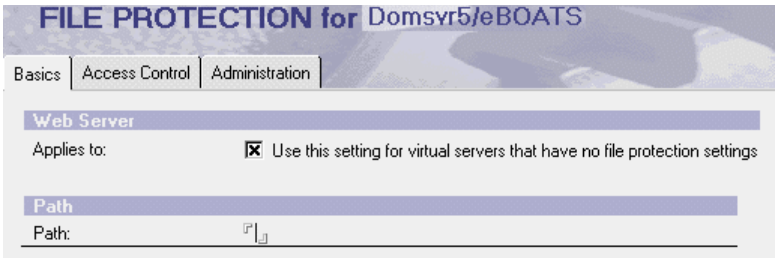


Figure 182. The File Protection document

5. In the Path section, type the directory path:

```
/lotus/domino/domsvr5/domino/servlet
```

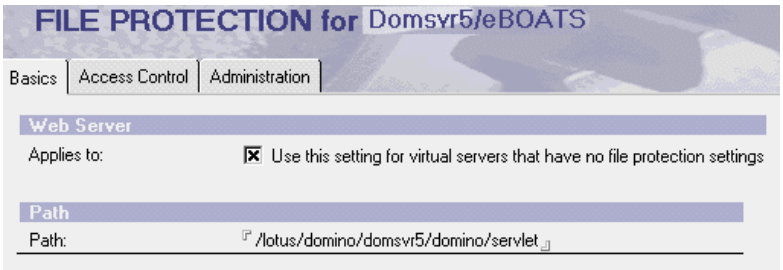


Figure 183. Path set in the File Protection document

6. When you click the **Access Control** tab, you get the panel shown in Figure 184.

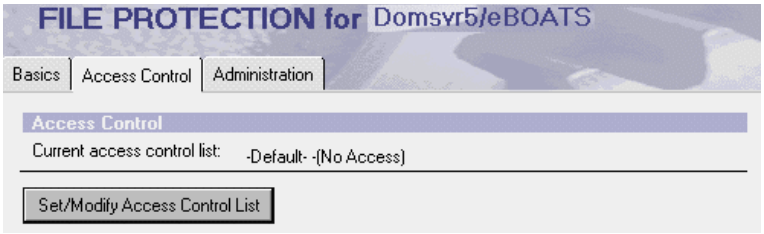


Figure 184. File Protection document access control tab

7. Click **Set/Modify Access Control List**. The window shown in Figure 185 appears. The only selection is `-Default- -(No Access)`.

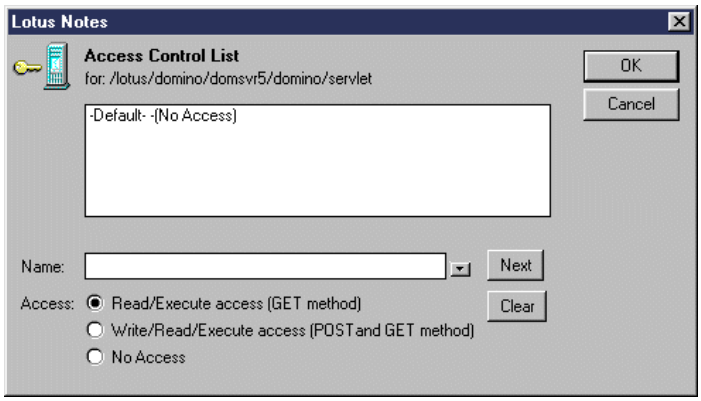


Figure 185. File Protection document access control list

8. Click the arrow to select a name. Choose **Employees**, and select the level access **Write/Read/Execute Access (POST and GET method)**. Then click **Next** (to get the Employees entry in the list). Click **OK**.
9. When clicking **Web Server Configurations**, you now see an entry for Business Partners, as shown in Figure 186 on page 262.

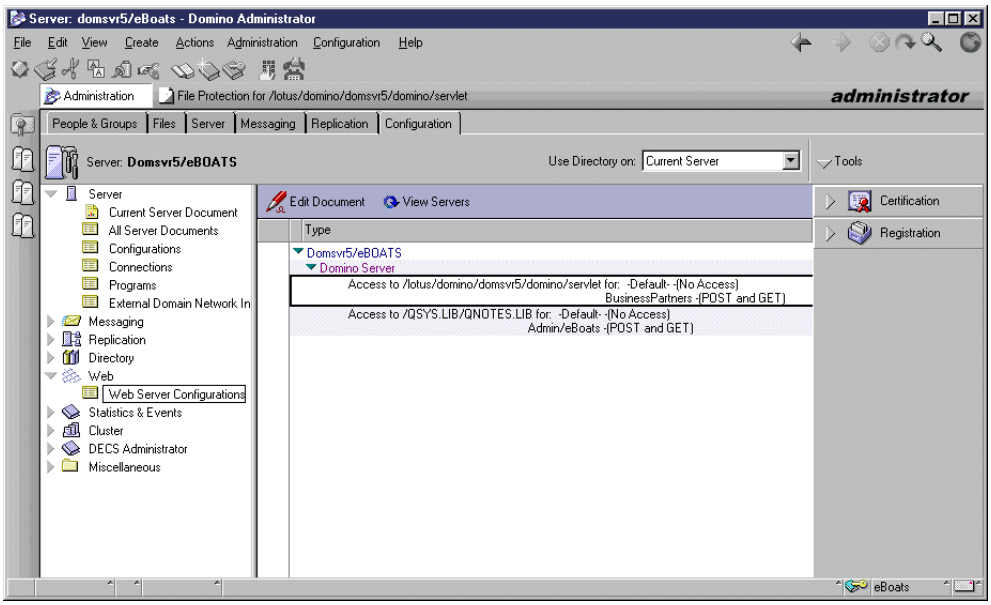


Figure 186. Web server configurations

10. Use the `tell http restart` command at the Domino server console.

11. To check that this change is taken into account, issue the `tell http show file access` command, described in 9.1.6, “Other useful HTTP console commands” on page 233. You should find the message: `Path protected: /lotus/domino/domsvr5/domino/servlet`

You need to be a business partner to run one of the servlets contained in the servlet directory. When entering the URL (for example on our server `http://domsvr5:8080/servlet/OrderQueryServlet`), you get a panel to enter a user profile and password, as shown in Figure 187.

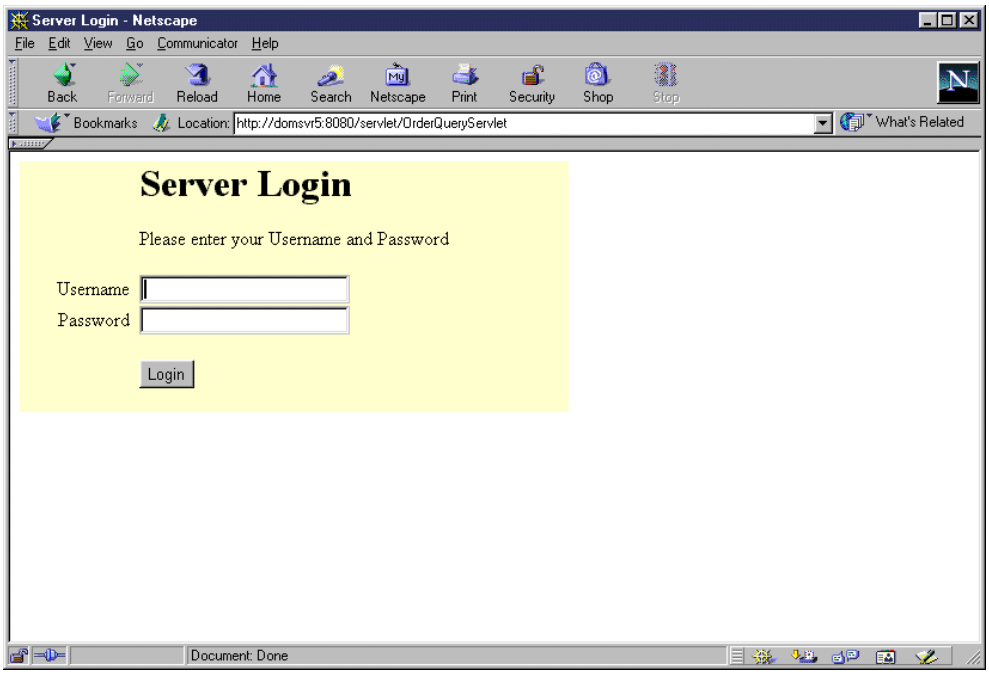


Figure 187. Accessing the servlet directly

After entering their user name and password, the user can access the page processed by the servlet, which has run without the cookie meant to provide the right identification.

The table displayed to the Web browser is empty and is of no use (Figure 188).

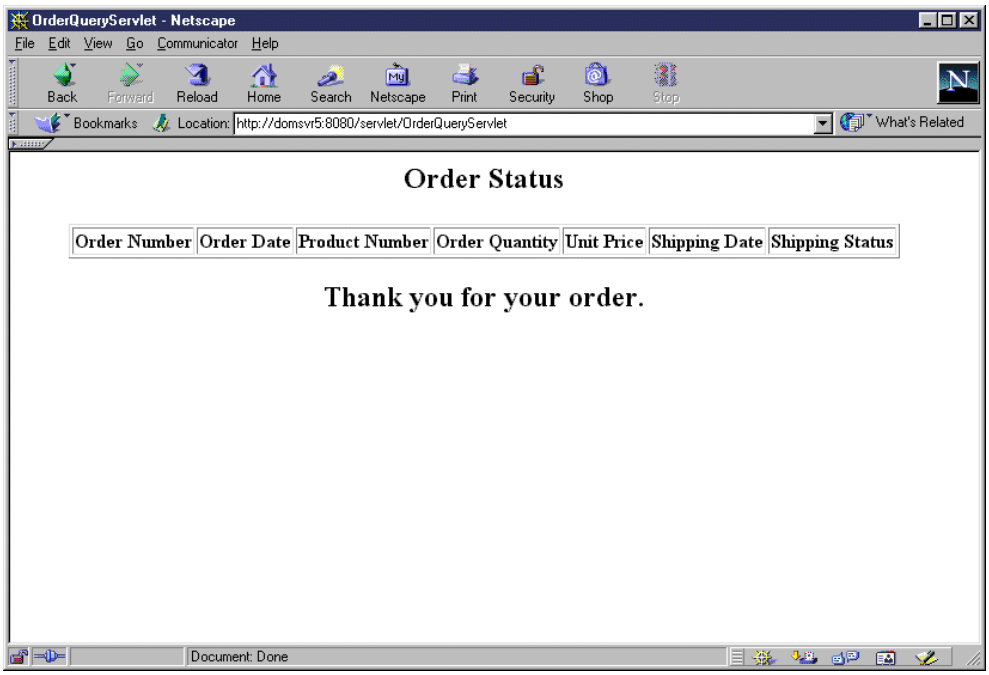


Figure 188. No relevant result for a servlet accessed directly

File protection versus SSL

In the eBoats International context, any access to servlets (or Domino databases) that need to be protected should be protected by:

- User authentication
- Encryption (using SSL)

There is no server setting for forcing SSL on servlets the way there is for Domino databases. One possibility is to have the servlet itself force a redirection if the connection is not SSL. This possibility is detailed in 11.4, “Enabling SSL at a servlet level” on page 302.

This is not enough for our context. User authentication is triggered before the servlet actually answers the request and forces SSL if required. If you look at the bottom left corner of Figure 187, you can see there is no padlock. This means that SSL is not forced when the user enters their user IDs and passwords, and the user ID and password are not encrypted before being transmitted to the server. This is not what we want.

In such a situation, our suggestion is to not implement authentication through file protection and force SSL by the servlet itself. Without authentication and the preceding logic to generate the necessary cookie, the servlet does not provide any sensible information anyway.

We highly recommend that you access the servlets through the regular application as demonstrated in 12.2, “Functions available for the company’s business partners (extranet)” on page 314.

Note

After you successfully complete the steps required in this chapter, you may decide to save the environment before going further in the security steps.

For recommendations about saving the Domino server and the LEI server, refer to 11.6, “Saving your Domino server configuration and environment” on page 303.

Chapter 11. Sample project SSL configuration

The security infrastructure that we are implementing covers the two scenarios shown in Figure 189 and Figure 190 on page 266.

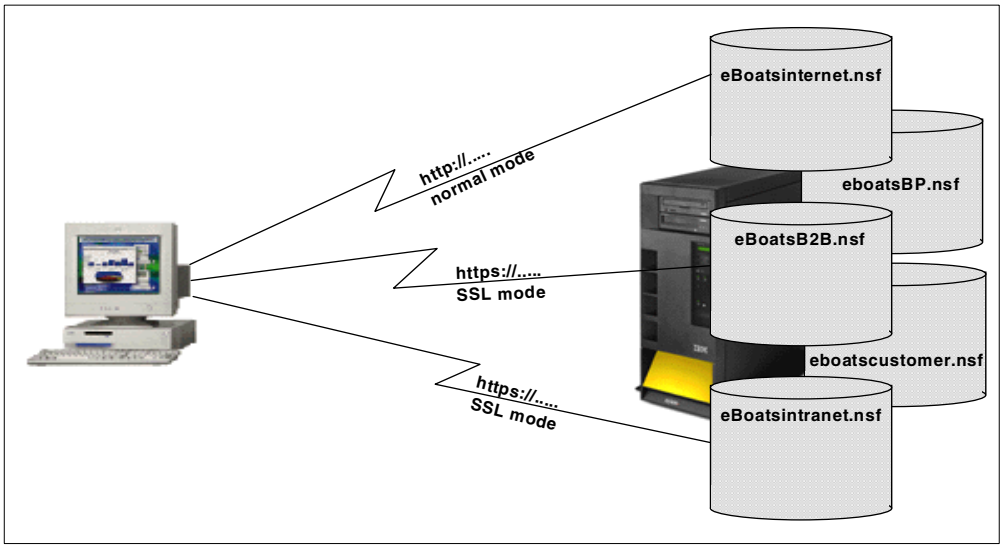


Figure 189. The normal mode and SSL mode implementation for eBoats International

Figure 189 shows which databases we want to connect to through http normal mode and which databases we want to connect to using the HTTPS SSL mode. The main structure is to connect to the eboatsinternet.nsf database through http normal mode by using the URL `http://systemname/eboatsinternet.nsf`. As mentioned earlier, we set the home URL to open the eboatsinternet database (eboatsinternet.nsf) so you have to write `http://systemname/`.

When we connect to the databases, eboatsB2B.nsf and eboatsIntranet.nsf, we use HTTPS SSL mode. Here we need to change to a secure connection for securing a user ID and password, as well as data being transferred.

To prevent users from connecting to the eboatsB2B.nsf and eboatsIntranet.nsf databases using the http normal mode, the property “Web Access: Require SSL Connection”. is chosen for both databases. This step is detailed later in 11.3, “Enabling SSL at a database level” on page 301.

11.1 Using digital certificates

In the eBoats International application, we are using certificates to secure the connections between the clients and server when requesting the databases, eboatsB2B.nsf and eboatsIntranet.nsf, as shown in Figure 189.

To accomplish this, we chose to establish the Domino server as a Certificate Authority (CA) to sign the certificates we need to support SSL server authentication.

When we establish the CA on the Domino server, we can request a signed certificate from this CA. The flow we are going through is shown Figure 190 on page 266.

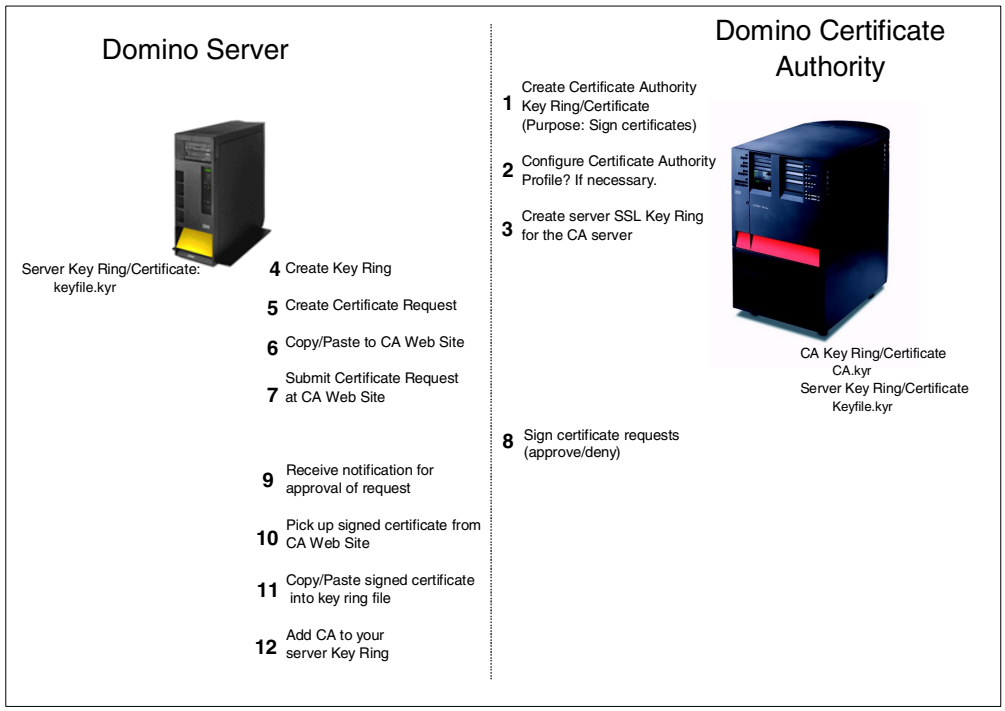


Figure 190. Setting up the use of certificates for SSL mode

In step 1 through 3, we establish the Domino server as being a Certificate Authority. Steps 4 through 7 create a key ring for the Domino server and create a certificate request to the CA, which we will copy and paste into the CA Web site for submission.

In step 8, we show how the CA receives the request for a certificate and how to approve or deny the certificate. When approval is being done, a notification is sent to the requester by e-mail.

In steps 9 through 12, we are back at the Domino server side (the requester) where we receive the notification from the CA. In the notification, a pick-up ID is supplied. We use this pick-up ID to go the CA Web site and sign in to receive our signed certificate. We copy and paste the certificate into our server key ring at our Domino server.

We also merge the CA certificate into our key ring. The is necessary for every SSL-enabled server to operate properly. Merging the CA certificate into the key ring of our Domino server provides the basis for successfully completing the initial SSL handshake to establish the trust relationship.

Note

If you are using one of the well-known CAs on the Internet, such as VeriSign, and the CA is provided in the list that automatically is added into your SSL key ring when you create it, this last step (step 12 in Figure 190) is not necessary.

11.1.1 Certificate Authority database setup

To establish the server to be a Certificate Authority, we need to set up the Certificate Authority database. In this database, you create and manage the special certificate that enables you to sign other certificates. You must use Domino and Notes R5 to use this application. For more information, see *Administering the Domino System or Domino 5 Administration Help* (help5_admin.nsf).

The Certificate Authority application lets you establish a Certification Authority (CA) within your organization. As an internal CA, you are responsible for handling certificate requests from Domino server administrators and clients within your organization and adding client certificates to the Domino Directory.

Use the Certificate Authority application to perform the following certification authority tasks:

- Create and manage the CA certificate and key ring file, which holds your CA certificate.
- Create the server certificate and key ring file for the CA server.
- Sign server certificates when server administrators request them so clients and servers can communicate.
- Sign client certificates when users request them to use client authentication and SMIME encryption and electronic signatures.
- Add client certificates issued by external CAs to the Domino Directory to set up client authentication on a server and to allow Notes clients to send encrypted SMIME mail messages.

Note

Which tasks can you perform if you're a Server administrator or client? Server administrators and clients use their browsers to access the Certificate Authority application. They use the application to:

- Submit requests for server or client certificates.
- Pick up server and client certificates.
- Install the CA's certificate as a trusted root.
- Submit a client certificate issued by an external CA to add to the server's Domino Directory for client authentication and encrypted S/MIME mail messages.

This is demonstrated later in 11.1.3, "Domino server certificate setup" on page 276.

To set up the database, perform the following steps:

1. Open your Notes workstation.
2. From the menu bar, select **File->Database->New**.

   The window shown in Figure 191 on page 268 appears.

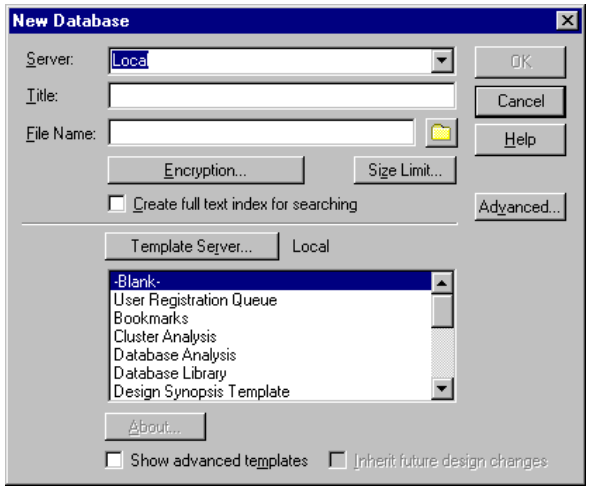


Figure 191. Creating a new database

3. Fill out the information as indicated in Table 33. The completed display is shown in Figure 192.

Table 33. Labels and values for opening the CA database

Label	Value
Server	The name of your Domino server (for example, Domsvr5/eBoats)
Title	Domino R5.0 CA
File name	certca.nsf
Template server	The name of your Domino server (for example, Domsvr5/eBoats)
Show advanced templates	Select. You need to select this before you can find the template, Domino R5.0 Certificate Authority (cca50.ntf), in the template list above.

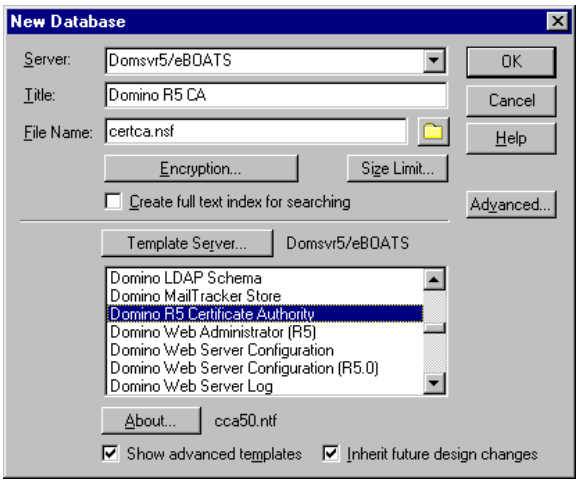


Figure 192. Creating a new database for the CA

4. Click the **OK** button.

This creates the Domino R5.0 Certificate Authority database. The About Certificate Authority document appears. Click **File->Close** two times, to return to your Notes client workspace.

Note: We experienced a problem trying to work directly in the database after closing the About Certificate Authority document. One or more error message boxes appear, with the messages `View or Navigator 'defView' does not exist` or `You have insufficient access to perform this operation.`

The database icon in Figure 193 appears on your Notes client workspace.

Figure 193. The Domino R5.0 Certificate Authority database

11.1.2 Certificate Authority setup

You are about to create a CA certificate, so you can sign server and client certificates, and add the CA's digital signature to the server and client certificates. The CA certificate is stored in a key ring file, which is a binary file that is password-protected.

When you use the following steps to create the CA key ring file, the file is stored by default in the client's Notes data directory. This means that the key ring by default will be placed on the C: drive on the workstation, for example, in the path c:\notes\data\CAKey.kyr.

For AS/400 users, an option is to map a drive to the AS/400 server that will point to the path where you want to store the key ring. For example, you can map drive H: to the AS/400 root directory, as shown in Figure 194.

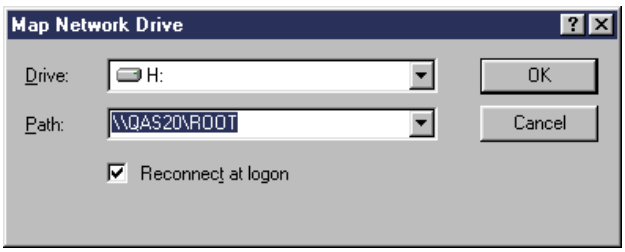


Figure 194. Mapping the Domino server data directory

To store the CA key ring file (and other key ring files generated later in the process) in the server's directory, the path that we will use is h:\lotus\domino\DOMSVR5\CAKey.kyr.

Note

To map drives to the AS/400 system, you need to configure and start the AS/400 NetServer.

AS/400 NetServer is an OS/400 function that enables Windows clients to connect to AS/400 shared directory paths and shared output queues using TCP/IP. It enables an AS/400 system to provide file and print serving in a Windows network without needing to install additional hardware or software on the AS/400 system. For more information about AS/400 Netserver, you can refer to the redbook *The AS/400 NetServer Advantage*, SG24-5196.

Double-click the database icon on the Notes client workspace.

This brings you into the Certificate Authority setup menu as shown in Figure 195. Make sure that the **Certificate Authority Configuration** is selected in the Navigator to the left.

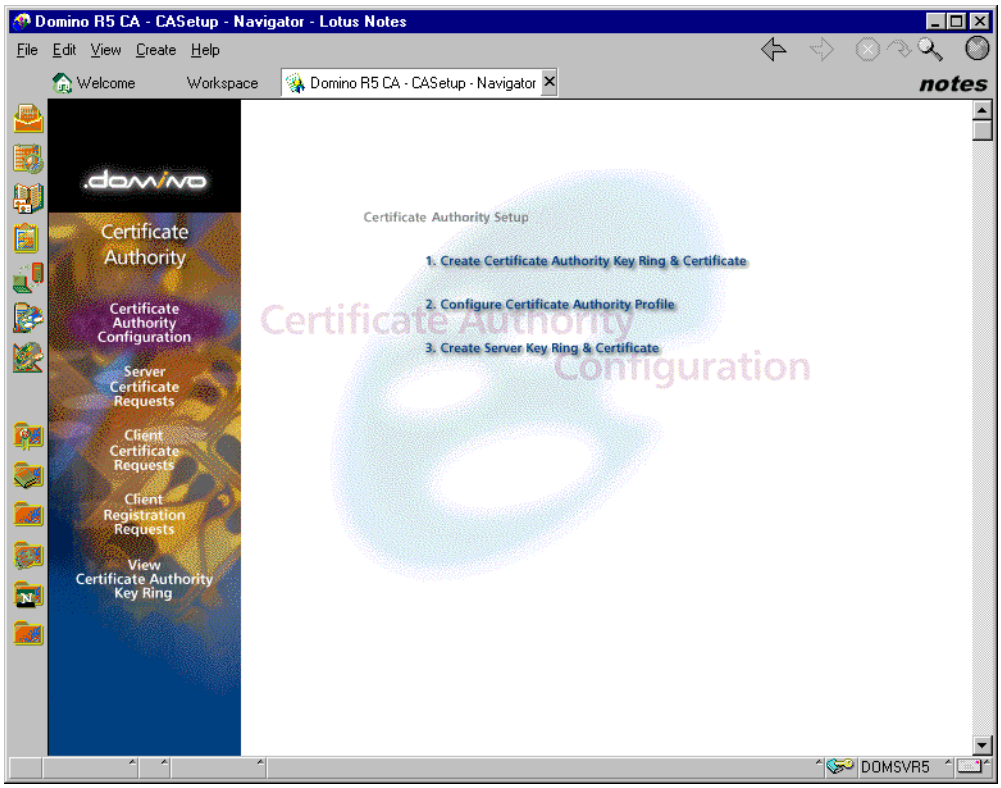


Figure 195. The Domino R5.0 CA setup menu

The first option, Create Certificate Authority Key Ring & Certificate, creates the CA's certificate and stores it in a password protected binary file. Perform the following steps:

1. Click menu option 1, **Create Certificate Authority Key Ring & Certificate**. This brings you to the window shown in Figure 196.

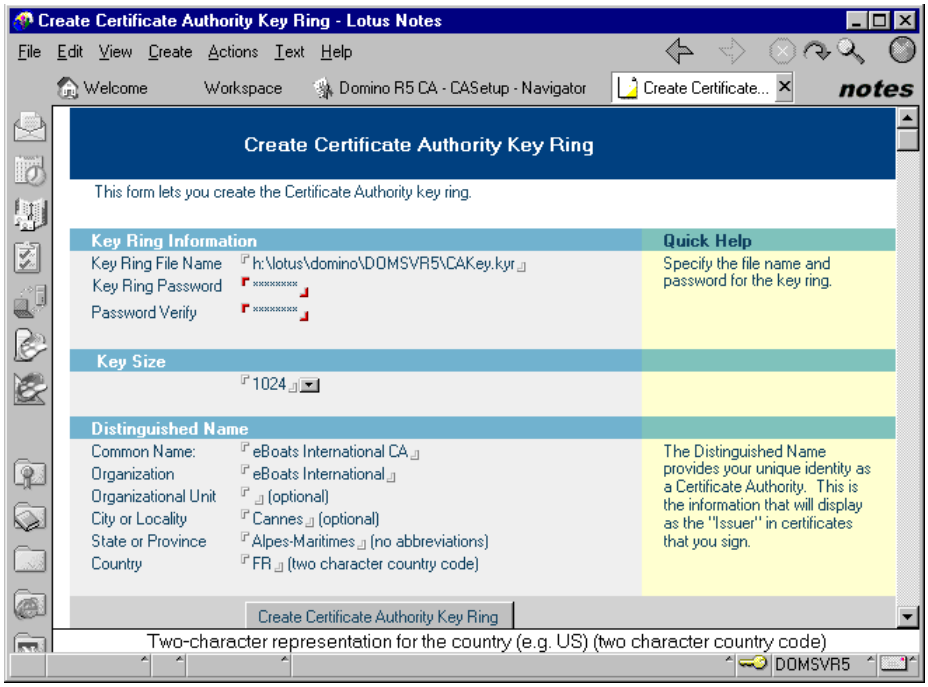


Figure 196. Create CA key ring and certificate

Enter the information that is required to define the CA key ring:

- **Key ring file name**: Specify the file name for the CA key ring. The default value is `CAKey.kyr`, but any name that will be easier for you to remember can be specified. For example, you can use the domain name for which the CA will sign certificates. This convention is especially helpful if you have more than one CA server and therefore more than one CA key ring.
- **Key ring password**: Specify and confirm the password for the CA key ring. You can specify any combination that meets your security guidelines. For security, choose a password that is different from the one you use when you are creating the key ring for your Domino Web server.
- **Key size**: Specify the key size in bits of the public/private key for the CA key ring. You can choose from 512 to 1024 (the default). You do not need the same key size for the SSL and CA key rings, but longer keys are to prefer for stronger security. The export regulations of the United States on key sizes for international use do not apply to the key sizes used in CA certificates.
- **Distinguished name**: The distinguished name provides your unique identity as a Certificate Authority. This is the information that displays as the “issuer” in certificates that you sign being the CA. Using all distinguished name fields decreases the chance of two servers having a name collision.

2. When you enter all the required information, click the **Create Certificate Authority Key Ring** button.
3. When you see the confirmation message (as shown in Figure 197) that your key ring has been created, click the **OK** button. You return to the set up menu shown in Figure 195 on page 270.

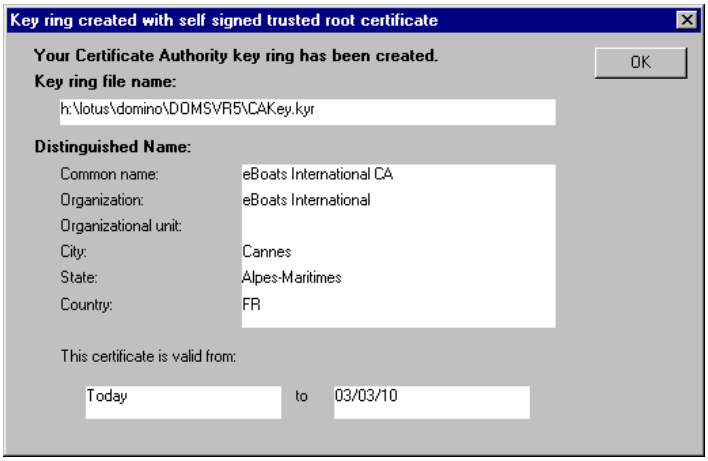


Figure 197. The confirmation message for creating the CA key ring

You have now created the Domino server Certificate Authority and can go on to create the SSL key ring for the CA.

Before we do that, we go through the second menu option 2, **Configure Certificate Authority Profile**, shown in Figure 195 on page 270. This option stores settings that the application needs to function properly.

Click the menu option. Then, you see the window shown in Figure 198.

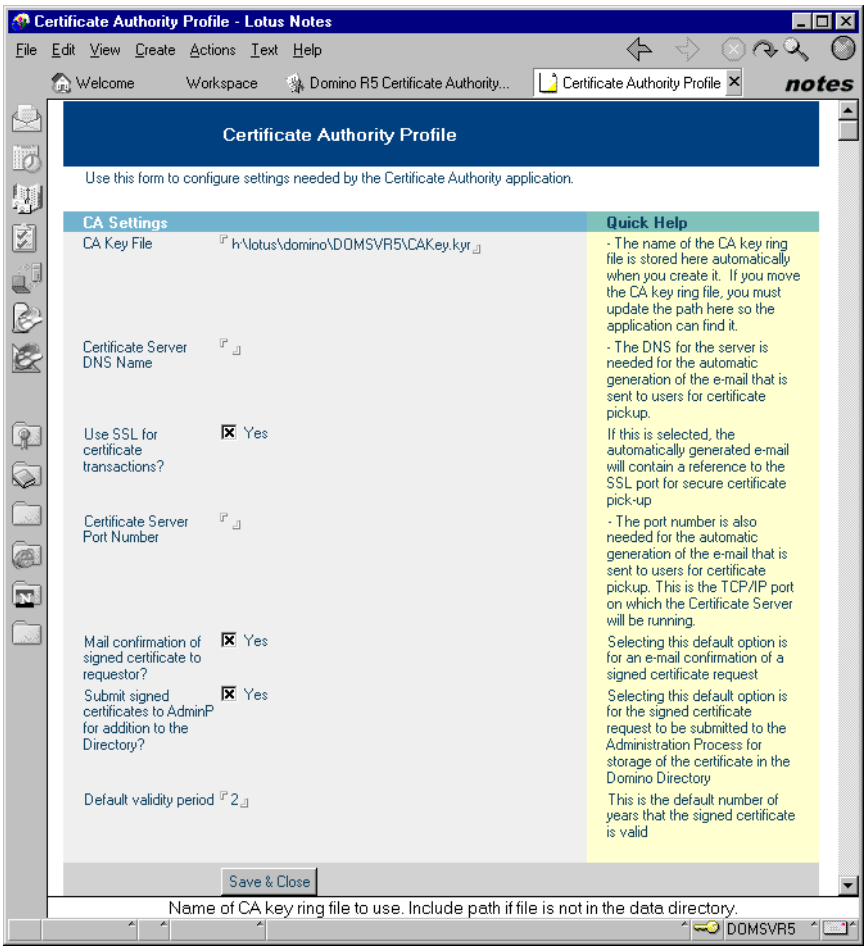


Figure 198. Certificate Authority Profile

The purpose of the CA profile is to set up a set of default values that will be used when signing certificates. These defaults are stored in the CA profile, which is also a part of the Certificate Authority application. You need to perform the following steps:

1. Go through the information required to configure Certificate Authority:

   - **CA Key file**: Specify the filename and location of the CA key ring. The default is the name and location you specified when you created the CA key ring. If you move the key ring to another location, you have to update the path here so the application can find it.
   - **Certificate server DNS name**: Specify the DNS host name (also referred to as the URL) of the certificate server. The DNS name is needed for the automatic generation of the e-mail that is sent to users for certificate pickup.
   - **Use SSL for certificate transactions**: Specify whether you want the server to use SSL for certificate transactions. The default is `Yes`. It can then be used when the e-mail is automatically generated. The e-mail will contain a reference to the SSL port for the secure certificate pickup.
   - **Certificate server port number**: The port number is also needed for the automatically generated e-mail that is being sent to the users for certificate pickup. This is the port on which the CA server is running. Enter the SSL

port number (the default is 443), if you enabled the previous parameter. Otherwise, enter the port number for the server running http normal mode (the default is 80).

- **Mail confirmation of signed certificate to requester**: By selecting this, your CA will automatically e-mail requestors when their certificates has been signed. The default is Yes. If you set it to No, you must set up your own manual mechanism of notification.
- **Submit signed certificates to AdminP for addition to the directory**: By selecting this, the signed certificate automatically is submitted to the Administration Process, which then puts them into the Adminp queue of certificates that must be added to the Domino directory. If you specify No, you must set up your own mechanism to perform these steps manually.
- **Default validity period**: Set the length of time in years for which you want to allow signed certificates to remain valid. The default is two years.

2. Click **Save & Close**.

Now we can move on to the last step for becoming a CA, which is to create the SSL server key ring and certificate for the CA server.

As mentioned before, all SSL-enabled servers must have their own SSL key ring to establish a secure connection. The CA is no exception. SSL communication from the CA is initiated by an SSL key ring that contains the signed certificate from the CA.

From the CA setup menu, shown in Figure 195 on page 270, select menu option 3, **Create Server Key Ring & Certificate**. The form displayed in Figure 199 appears. The following steps are necessary:

1. Enter the following information:
   - **Key ring file name**: Specify a name and a path for the SSL key ring file. The default is `keyfile.kyr`. Remember that you must have a mapped drive to the AS/400 system to accomplish this in an AS/400 environment. With a mapped drive, you can specify the name and the path, for example, h:\lotus\domino\DOMSVR5\CAkeyfile.kyr.
   - **Key ring password**: Specify and confirm the key ring password. The considerations described earlier when creating the CA key ring and certificate also apply to this parameter.
   - **Key size**: Specify the size in bits of the public/private key for the SSL key ring. You can choose between 512 bits, which is the default, or 1024 bits. Be aware that longer keys provide stronger encryption. However, United State export regulations limit your choice of SSL key size to 512 bits for international use. For eBoats International, we chose 512 bits.

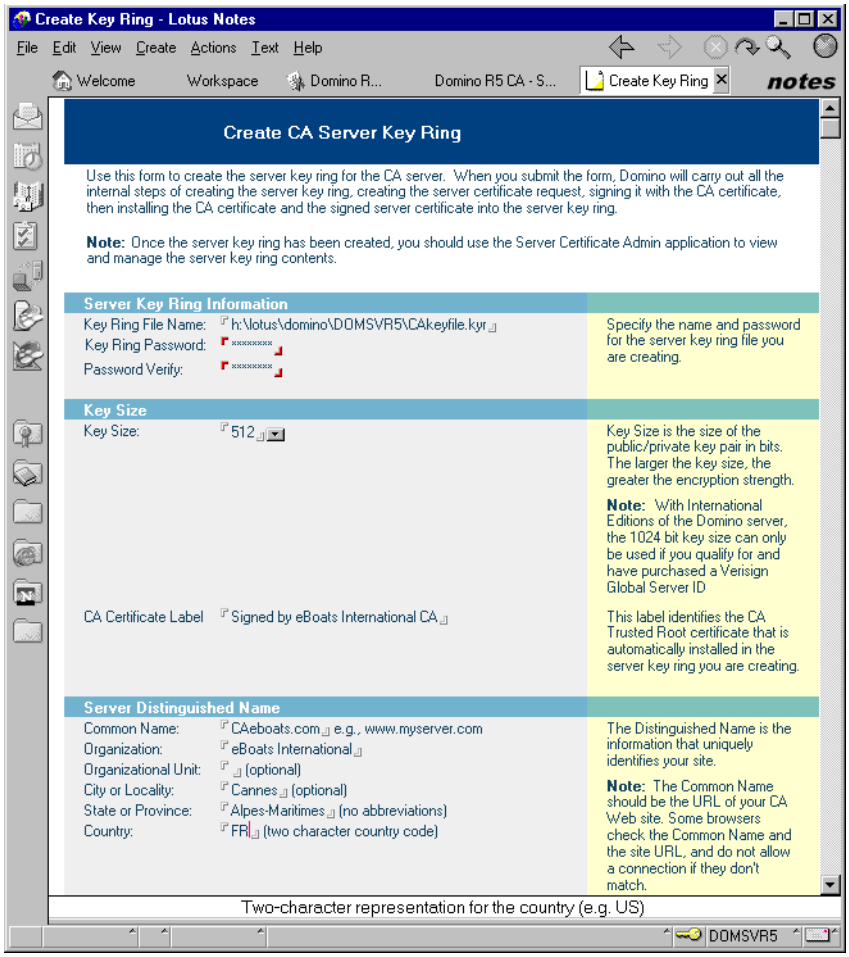


Figure 199. Create CA Server Key Ring

- **CA certificate label**: The label you specify will be used to identify the CA as a trusted root in the SSL key ring in eBoats International.
- **Distinguished name**: The distinguished name identifies your CA server, which is where users will request and pickup certificates. The common name field will be populated with the DNS host name for your CA Web site. For some browsers, the common name in the certificate will be compared to the host name in the URL being requested. If there is a mismatch, a warning is shown.

2. Click the **Create Server Key Ring** button. You are presented with the Enter Password window for the CA key ring as shown in Figure 200 on page 276.

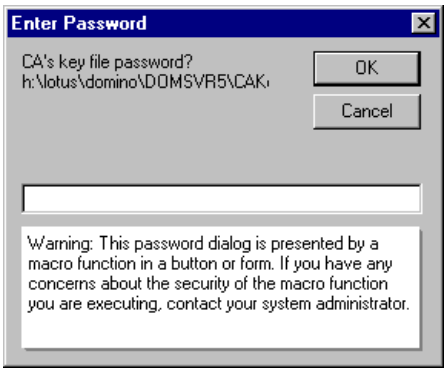


Figure 200. Enter Password for the CA key ring

3. Enter the password for the CA key ring. The CA key ring is the key ring you created when you selected earlier menu option 1, Create Certificate Authority Key Ring & Certificate.
4. Click the **OK** button. You see the confirmation window shown in Figure 201.

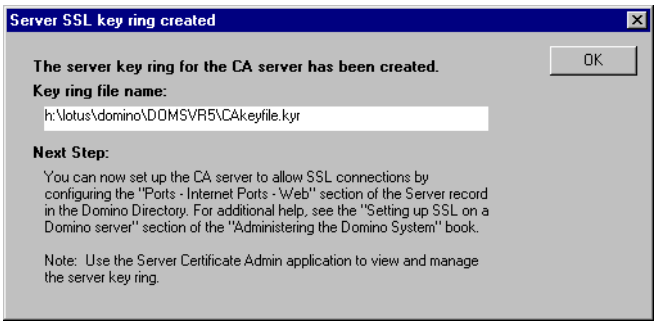


Figure 201. Server SSL key ring created confirmation

5. Click the **OK** button, and you return to the CA setup menu.

Having done this, you are ready to begin being a CA.

11.1.3 Domino server certificate setup

Next, we must create a key ring and certificate to be used by eBoats International Domino Web server. To do this, we use the Server Certificate Admin application. Domino automatically creates the Server Certificate Admin application during server setup.

The Server Certificate Admin application lets you request server certificates from either an internal or external Certification Authority (CA) and manage your server certificates in a key ring file. You must access the application using a Notes client or the Domino Administrator (R5.0). You must also have a Web browser available on the client to retrieve certificates from the Certificate Authority server.

The Server Certificate Admin application allows you to:

- Create the server certificate and key ring file, which holds your server certificate.
- Send a request to a CA to sign the server certificate.
- View requests that you submitted to CAs.

- Add a CAs certificate as a trusted root to the server certificate.
- View information about certificates in the key ring file.
- Control client access to the server by adding or removing trusted root certificates from the key ring file.
- Create a self-certified certificate for testing purposes.

If the Server Certificate Admin application is not available after you start the Domino server, use the Server Certificate Admin template (CSRV50.NTF) to create it:

1. Open your Notes workstation.
2. From the menu, bar select **File->Database->New**.

   The window shown in Figure 202 appears.

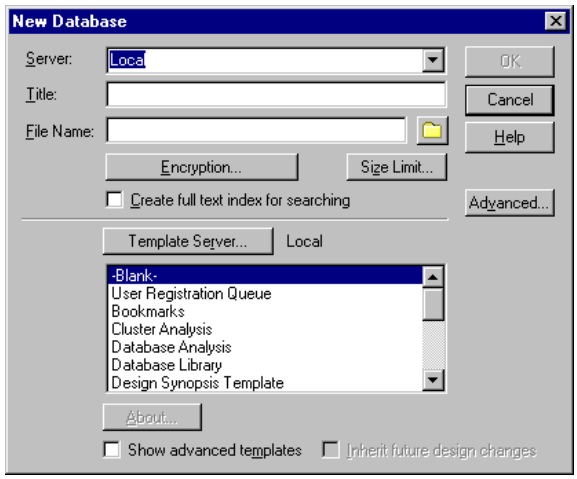


Figure 202. Creating a new database

3. Fill out the following information as indicated in Table 34. The completed display is shown in Figure 203 on page 278.

Table 34. Labels and values for opening the server certificate admin database

Label	Value
Server	The name of your Domino server (for example, Domsvr5/eBoats)
Title	Server Certificate Admin
File name	certsrv.nsf
Template server	The name of your Domino server (for example, Domsvr5/eBoats)
Show advanced templates	Select. You need to select this before you can find the template, `Server Certificate Admin, csrv50.ntf`, in the template list above

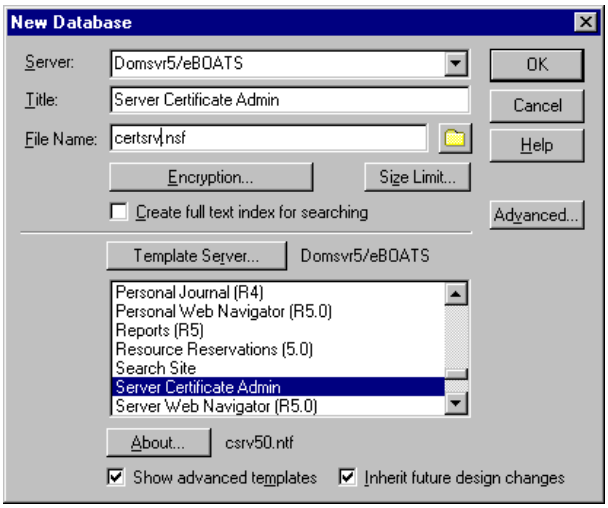


Figure 203. Create new database for the server certificate admin database

4. Click the **OK** button.

This creates the Server Certificate Admin database. The database icon (Figure 204) appears on your Notes client workspace.

Figure 204. The Domino R5.0 Certificate Authority database

5. Double-click the database icon on the Notes client workspace.

This brings you into the Server Certificate Admin setup menu as shown in Figure 205. Make sure that Create Key Rings and Certificates is selected in the navigator on the left side of the display.

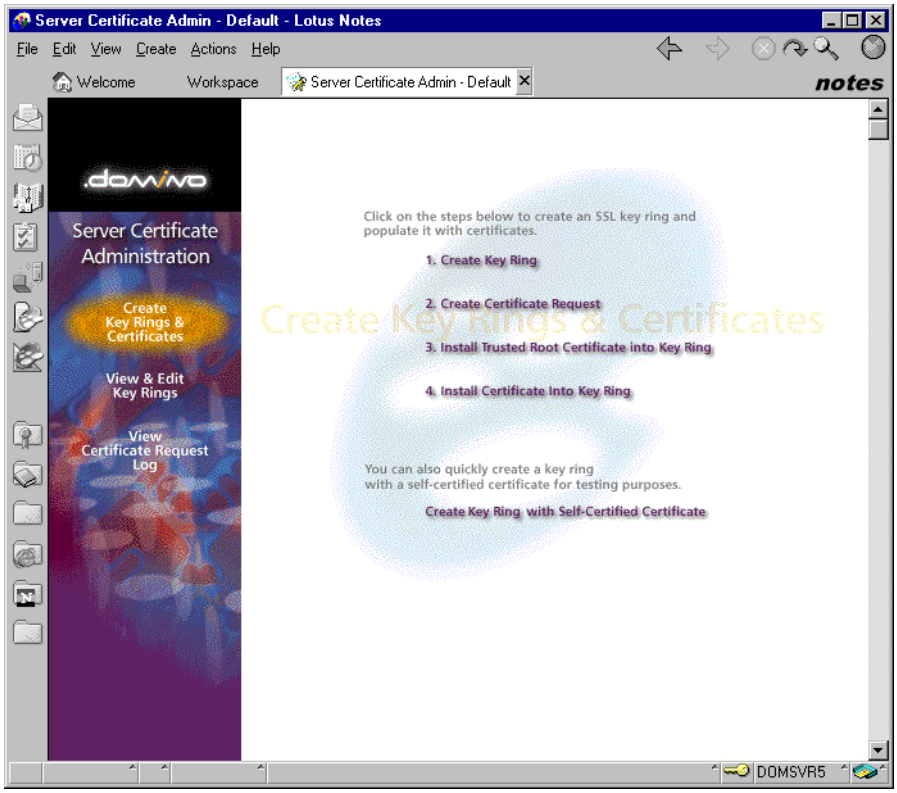


Figure 205. The Domino Server Certificate Admin menu

11.1.3.1 Create Key Ring

Option 1, Create Key Ring, creates a password-protected file that holds the server certificate and trusted roots. To use it, perform the following steps:

1. Click menu option 1, **Create Key Ring**. This brings you to the window shown in Figure 206 on page 280.
2. Type in the following information:
   - **Key ring file name**: Enter the filename for the SSL key ring. By default the filename is set to `keyfile.kyr`. The Domino server document also contains the default name under the Ports - Internet ports tab. Although it is good practice to use the default name, you can choose to give the key file another name. If you are having the CA and your Domino Web server running under the same Domino server, the name of the CA SSL key ring and your server SSL key ring will have the same name. You may find it easier to call the two SSL key rings different names. If you choose to call the key ring another name, remember to change the server document correspondingly.
   - **Key ring password**: Specify and confirm the password for the SSL key ring. You can use any combination of letters and numbers that meets your security guidelines.
   - **Key size**: Specify the size in bits of the public/private key for the SSL key ring. You can choose between 512 bits, which is the default, or 1024 bits. Be aware that longer keys provide stronger encryption. However, United State export regulations limit your choice of SSL key size to 512 bits for international use. For eBoats International, we chose 512 bits.

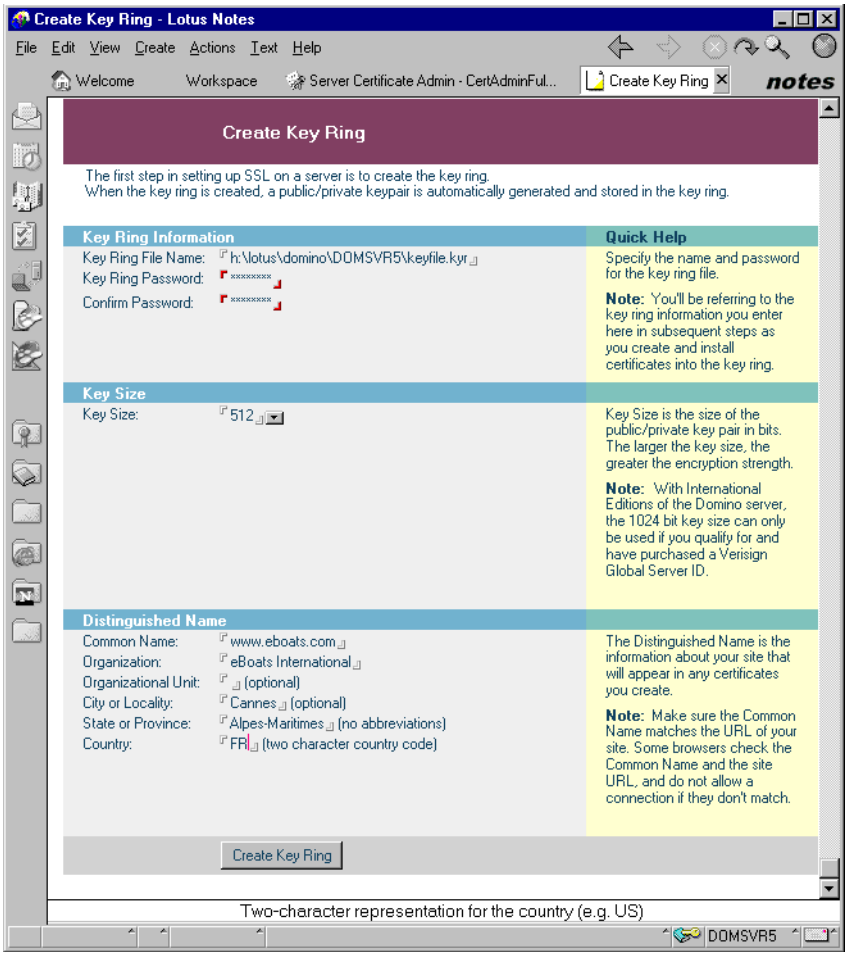


Figure 206. Create server key ring

- **Distinguished name**: The distinguished name identifies your CA server, which is where users will request and pickup certificates. The common name field will be populated with the DNS hostname for your CA Web site. For some browsers, the common name in the certificate will be compared to the hostname in the URL being requested. If there is a mismatch, a security warning is shown.

3. Click the **Create Key Ring** button. When you see the confirmation message shown in Figure 207, your key ring has been created.

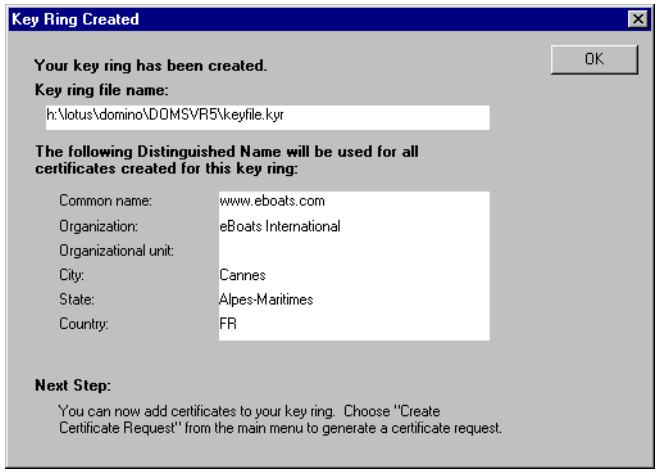


Figure 207. The create key ring confirmation message

4. Click the **OK** button to return to the Server Certificate Admin setup menu (as shown in Figure 205 on page 279).
5. If you want to see the key ring you just created or any of the public CAs that Domino automatically adds to your key ring as trusted roots, click **View & Edit Key Rings** in the Navigator menu on the left.

This process creates an SSL key ring that contains an *unsigned* certificate for the server. Therefore, the server is not yet ready for SSL. The certificate is incomplete until it is signed by a trusted entity (like a CA).

Note

It is possible to create a certificate called a *self-certified certificate*. This option is useful for the purpose of testing. Getting a certificate signed can sometimes introduce a delay. Therefore, a self-certified certificate is a useful shortcut if you want to begin testing immediately. To do this, you can use the menu option `Create Key Ring with Self-Certified Certificate` from the Server Certificate Admin setup menu.

The created key ring and self-signed certificate allow untrusted SSL sessions to be established between clients and servers. These sessions are encrypted, but not authenticated. It is extremely risky to use this option for anything other than testing purposes.

11.1.3.2 Create Certificate Request

We can now prepare to create the *certificate request* to the Certificate Authority. The Create Certificate Request option creates an unsigned certificate and several default trusted root certificates. It also lets you submit this request to an internal or external CA for signing.

In this section, we not only see the request creation, but also:

- The request submission to the CA
- The approval by the CA
- The approval notification by the CA

Creating the certificate request

To create the certificate request, perform the following steps:

1. From the menu displayed in Figure 205, select menu option 2, **Create Certificate Request**, from the Server Certificate Admin setup menu. The display in Figure 208 appears.

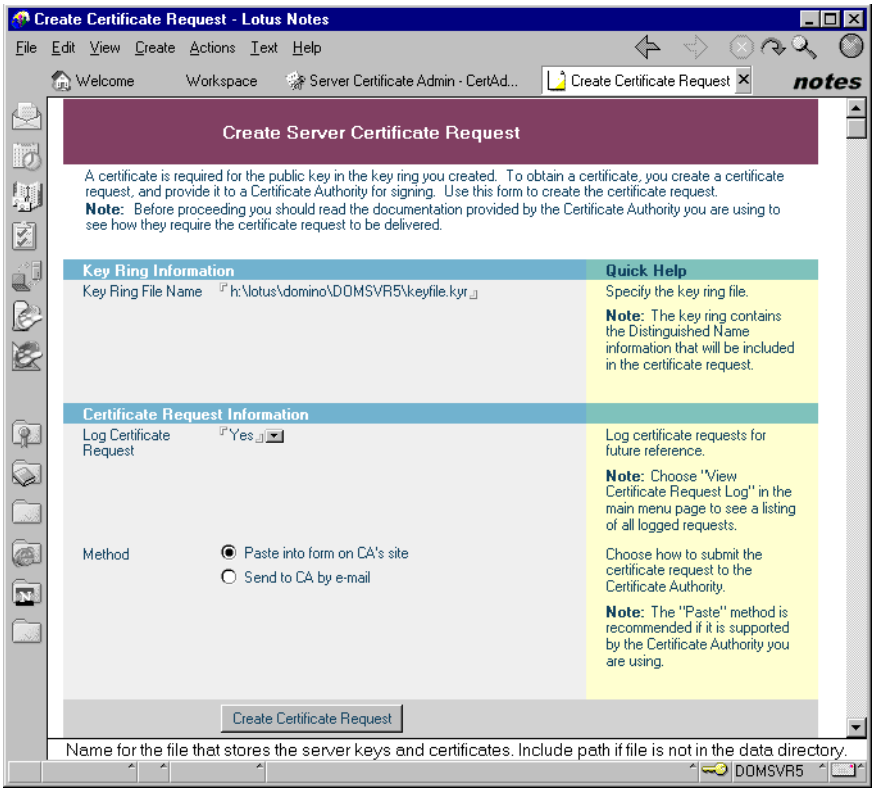


Figure 208. Create Server Certificate Request

2. Specify the SSL key ring for which you want to create the certificate request. It defaults to the filename you specified when you created the SSL key ring.
3. Choose whether you want to log the request. While you generally have only one request per key ring, request logging is a good administration practice. The Server Certificate Admin application includes a view to track your certificate request. In eBoats International, we chose Yes.
4. Choose how you want to submit the request to the CA. You can either copy and paste it into a form through a Web interface or send it to the intended CA through e-mail. If you choose the e-mail option, enter the e-mail details for yourself and the recipient. The appropriate choice depends on what is offered by the CA. Typically public CAs prefer the submit through the Web method. For eBoats, we chose Copy/Paste.
5. Click the **Create Certificate Request** button.
6. When the password prompt display appears, as shown in Figure 209, type the password for your server SSL key ring file.

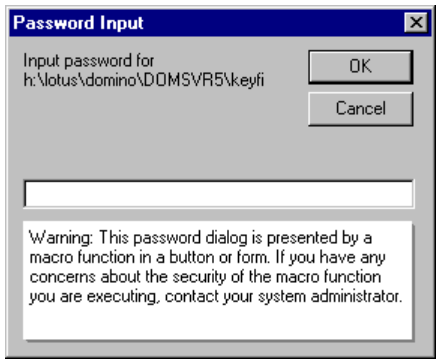


Figure 209. The key ring password prompt

7. Click the **OK** button. You see a confirmation message showing you the created request as shown in Figure 210.

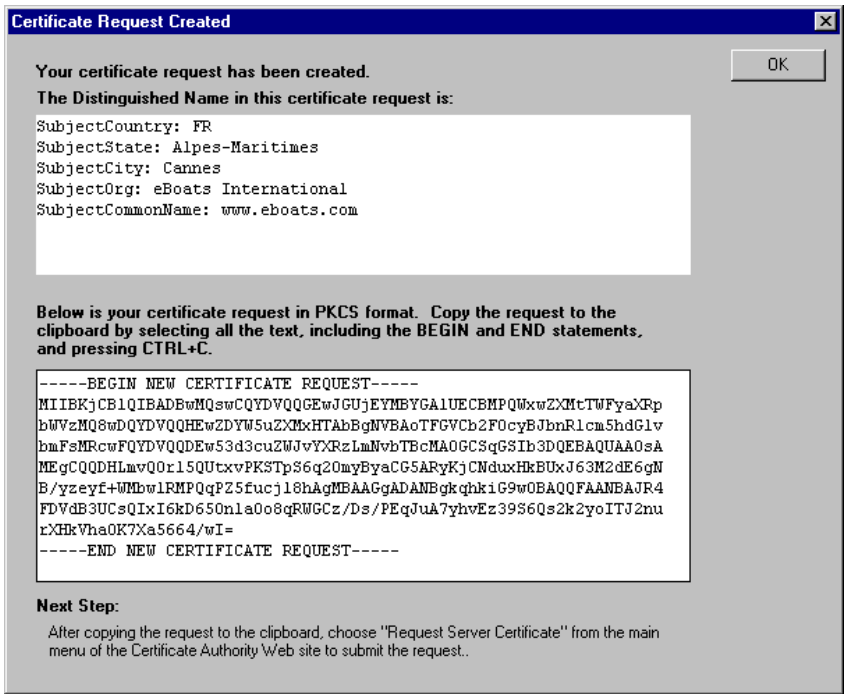


Figure 210. The certificate request confirmation message

The cursor is positioned at the start of the field that contains the PKCS10-format certificate request.

8. Highlight the entire contents of the field, *including the Begin and End statement lines*. Then, copy it to the clipboard using the keyboard, as explained in the help text, in the middle of the panel.
9. Click the **OK** button. You return to the Server Certificate Admin setup menu.

You are about to request a server certificate.

Note

You can request and obtain a server certificate from either a Domino or third-party CA. A server certificate is a binary file that uniquely identifies the server. The server certificate contains a public key, a name, an expiration date, and a digital signature. It is stored on the server's hard drive.

When you request an SSL server certificate, you use standard Public-Key Cryptography Standards (PKCS) format, an industry-standard format that many CAs, including Domino, understand. Before you request a certificate from a third-party CA, make sure the CA uses the PKCS format, not some other format, such as Privacy-Enhanced Mail (PEM). If you are unsure of the format required by a third-party CA, check with the CA.

Submitting the request to the CA

To submit this request to the Certificate Authority, open you browser, and go to the Web site of the CA that you selected.

For eBoats, we chose the CA we created earlier. The URL is `http://Domsvr5:8080/certca.nsf`. This takes you to the page shown in Figure 211.

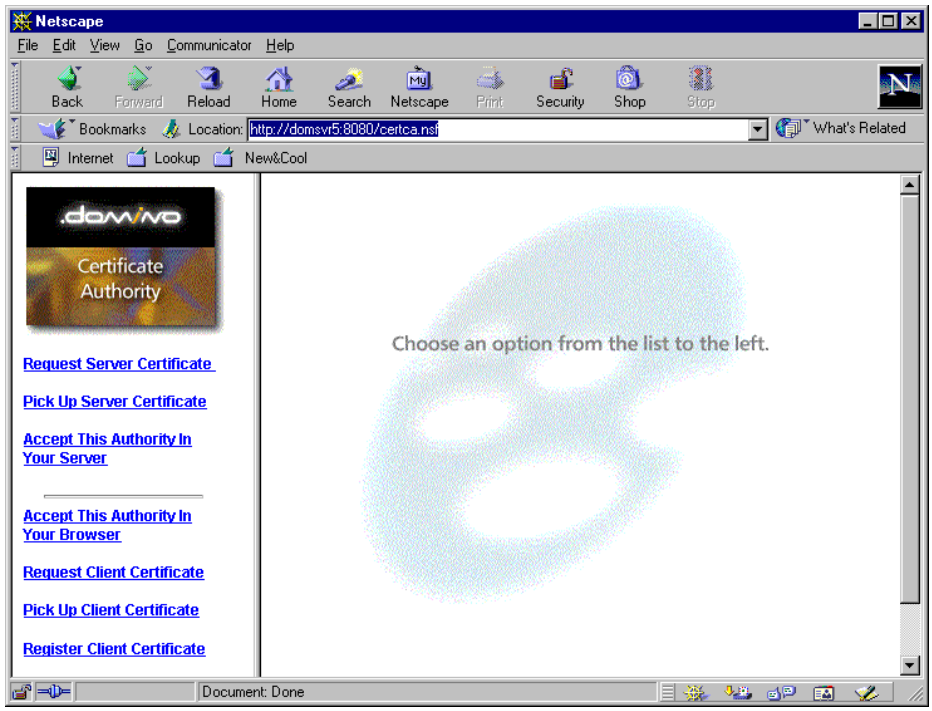


Figure 211. The CA Web site

Complete the following steps:

1. Click **Request Server Certificate**.
2. Enter the information required to identify the administrator, in the top of the form shown in Figure 212.
3. Move the cursor to the appropriate field, and paste the certificate request from the clipboard into the Certificate Request box.

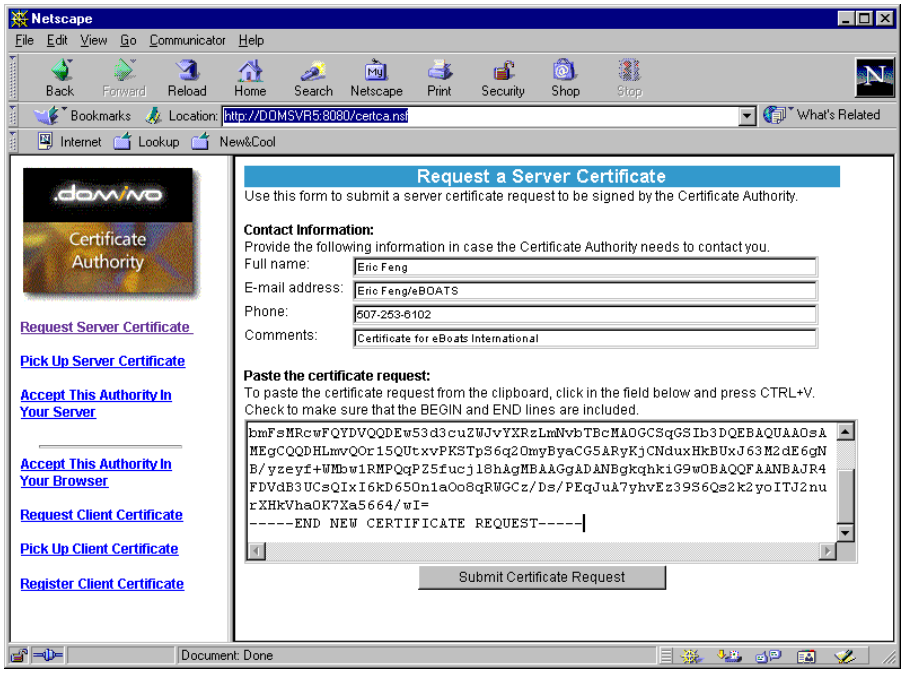


Figure 212. Request a Server Certificate

4. Click the **Submit Certificate Request** button.

   When the CA server has acknowledged your request into its queue, your browser shows a confirmation display (Figure 213).

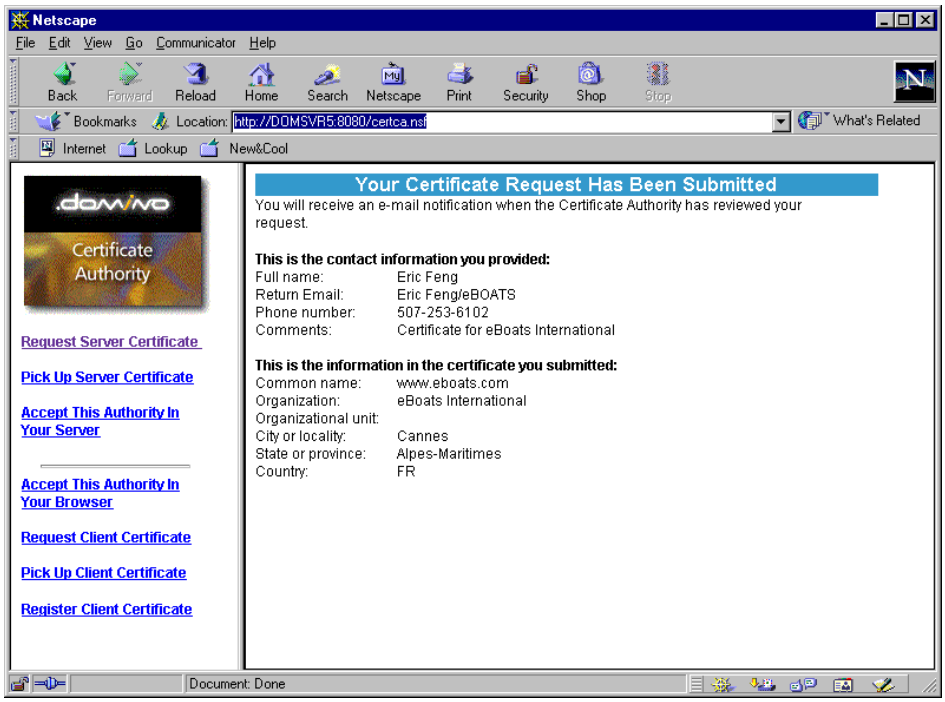


Figure 213. Certificate request confirmation

5. Go back to the Server Certificate Admin application, and close the dialog box that contains your certificate request.

You now have to wait until the CA approves your certificate request and sends you a notification on how to pick up the signed certificate.

Request approval by the CA

Being our own CA, we can simulate the process that has to be done by the trusted CA through this process:

1. Go to the Domino R5.0 CA application as shown in Figure 214.
2. Click the **Server Certificate Request** in the Navigator menu.

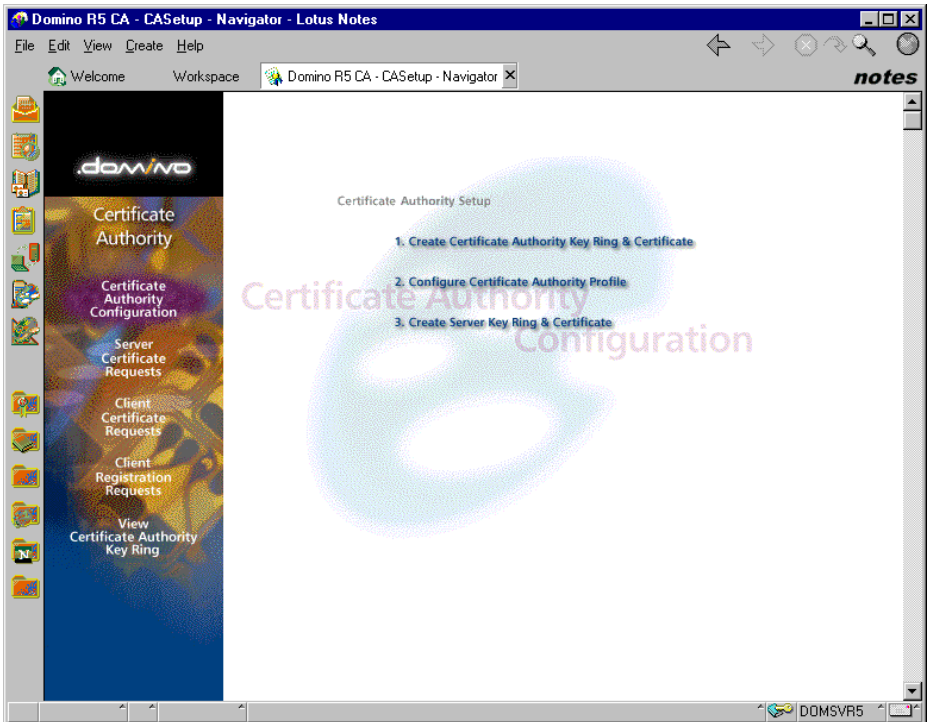


Figure 214. The CA setup menu

3. From the list of certificate requests waiting for approval, choose the request that you submitted (see Figure 215).

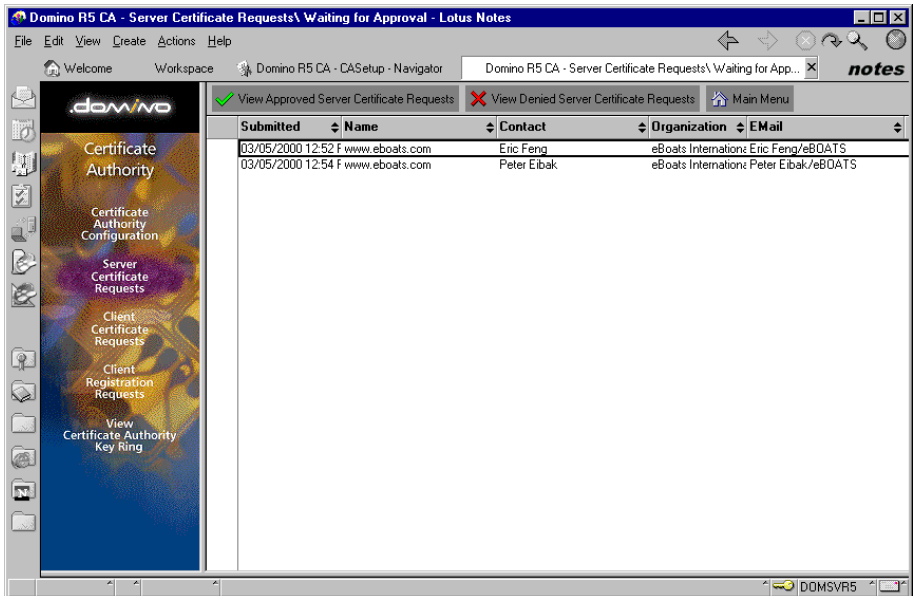


Figure 215. List of certificate requests waiting for approval

4. Click the request. The display in Figure 216 on page 288 appears.

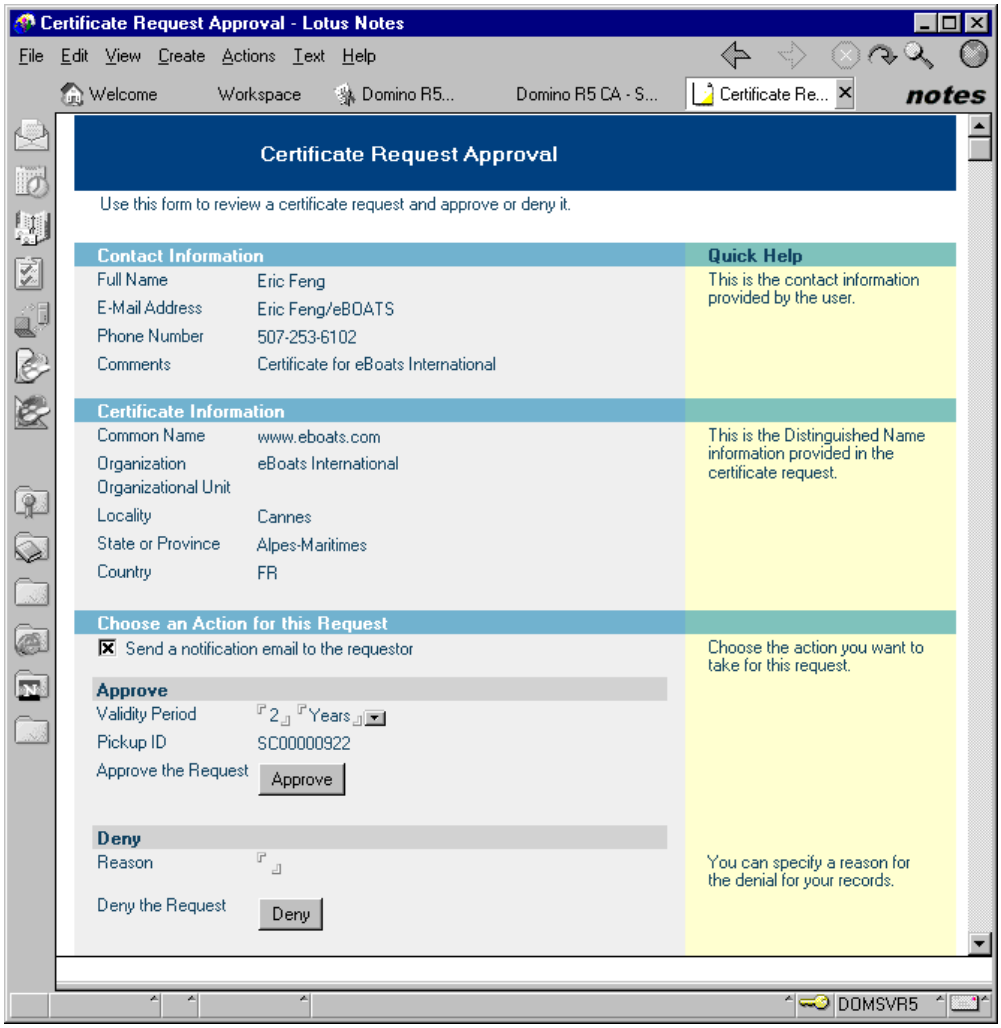


Figure 216. Certificate waiting for approval

5. Modify the validity period (if the default value, which is 2 years, does not fit).
6. Approve the certificate request by clicking the **Approve** button.
7. You are prompted for the password for the CA key ring, which is the first key ring we created for establishing the CA, as shown in Figure 217.

Figure 217. The CA key ring password prompt

Note: You may also be prompted for a DNS server name. Don't type anything, unless your network environment requires it.

8. You return to the list of certificate request waiting for approval, as shown in Figure 218.

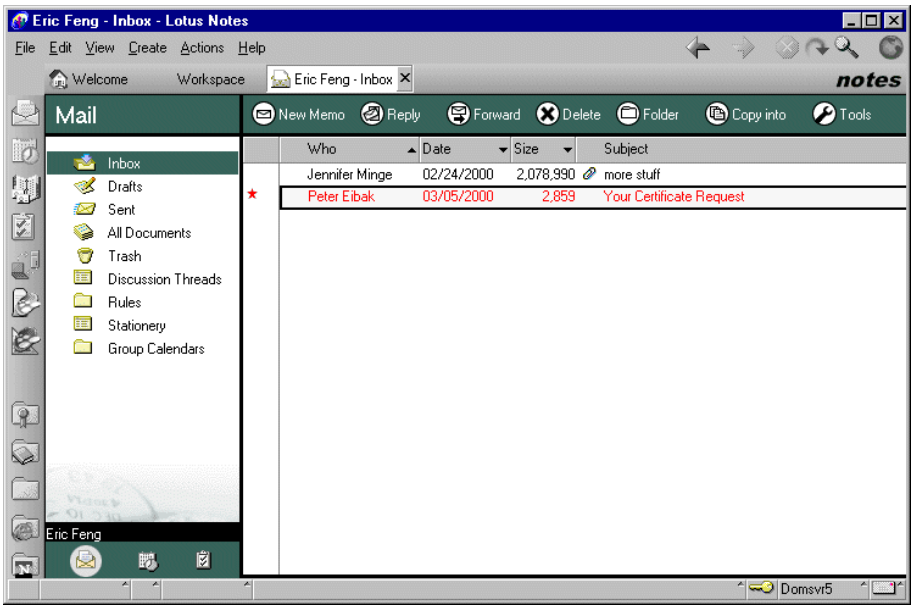


Figure 218. List of certificate requests waiting for approval

Request approval notification reception

A notification is now sent to the administrator that submitted the Certificate request. The next step is for the administrator to check their mailbox in Domino to see if a notification has arrived. The notification will look like the one shown in Figure 219.

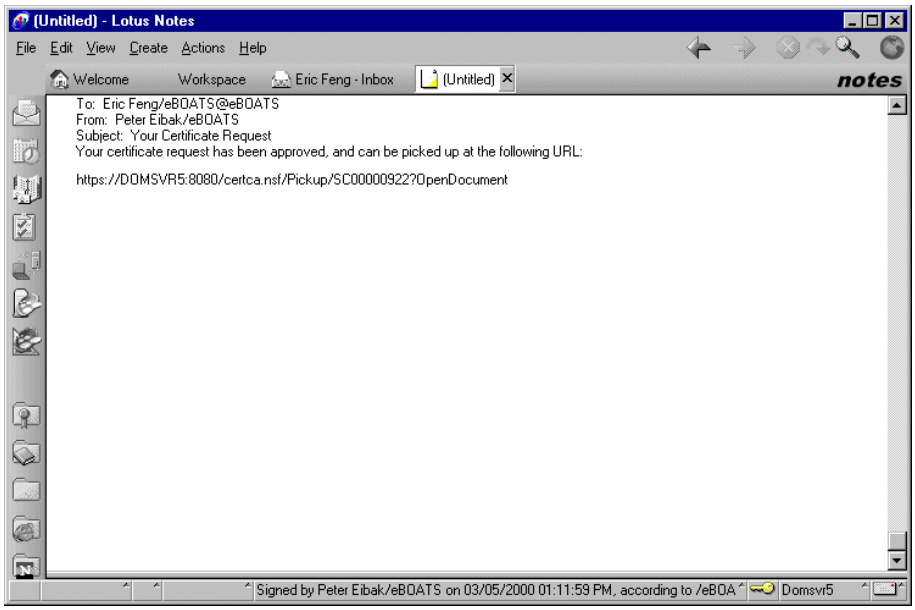


Figure 219. The notification received by the administrator

In the notification, a URL is specified for picking up the signed certificate. This URL was automatically created on the basis of the Certificate Authority Profile we configured earlier.

11.1.3.3 Accepting the CA authority in the server

In the URL, a pick-up ID is given. To use it, perform the following steps:

1. Write it down, and go to the Web site for the CA by using the URL (in our example `http://Domsvr5:8080/certca.nsf`). Figure 220 shows the welcome page of the site.

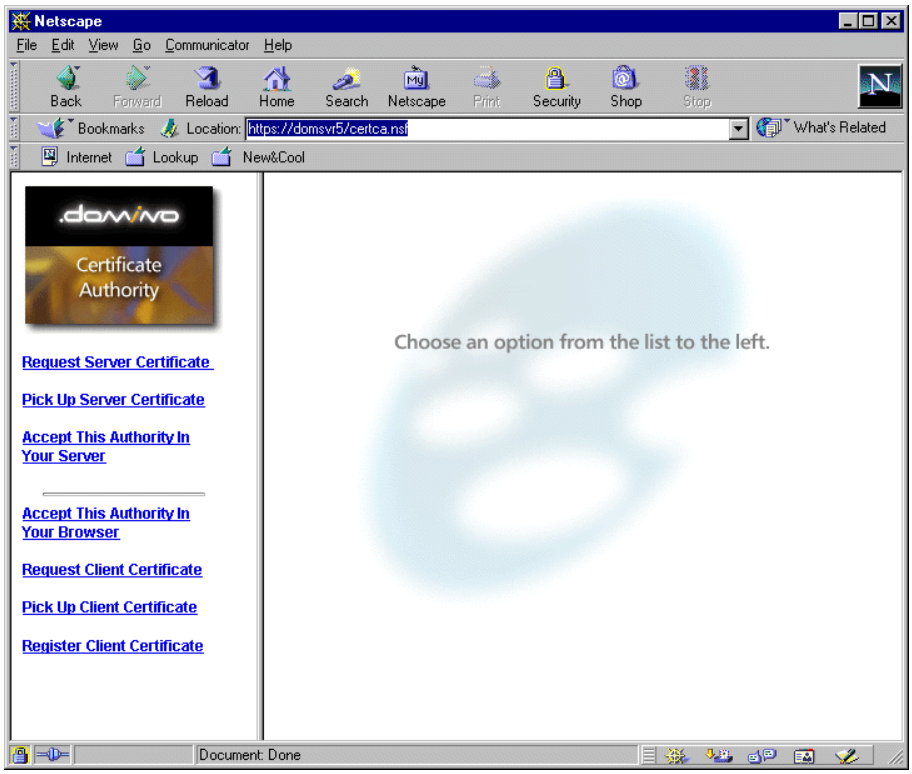


Figure 220. The CA Web site

2. Click **Accept This Authority in Your Server**. Figure 221 shows the next panel that appears.

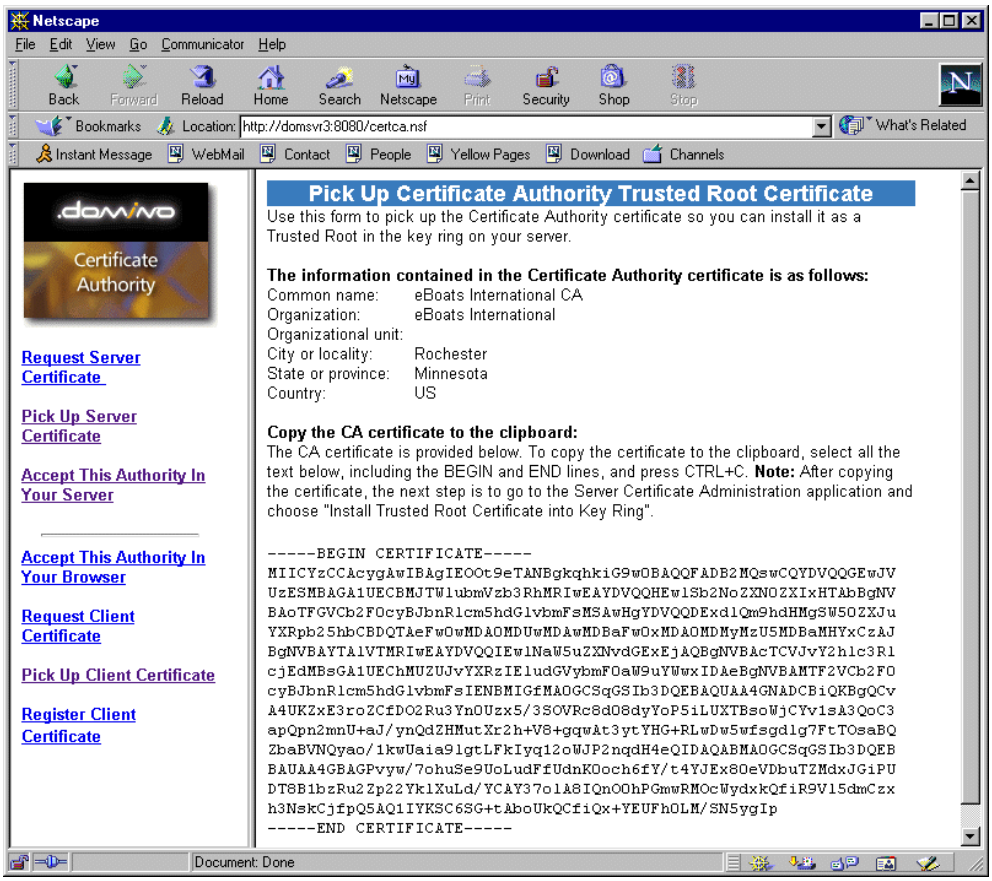


Figure 221. Pick up CA trusted root certificate display

3. Copy the certificate to the clipboard by selecting all the text below, including the BEGIN and END lines, and press CTRL+C.
4. Switch from the Web browser to your Lotus Notes client.
5. Open the Server Certificate Admin application from the Notes workspace. The welcome panel is shown in Figure 205 on page 279.
6. Click 3, **Install Trusted Root Certificate into Key Ring**. The document shown in Figure 222 on page 292 appears.

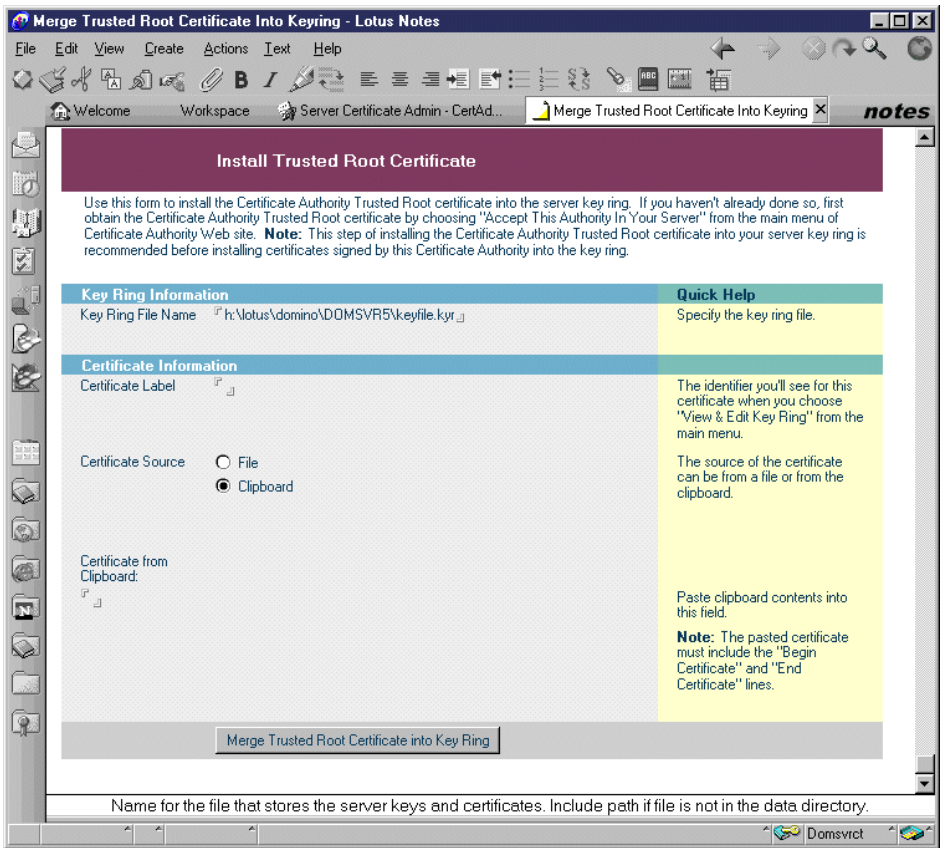


Figure 222. Install Trusted Root Certificate document

7. Type a certificate label.
8. Select **Clipboard** as the source for the certificate.
9. Paste what you copied from the Pick up CA trusted root certificate display in the Certificate from Keyboard field, as shown in Figure 223.

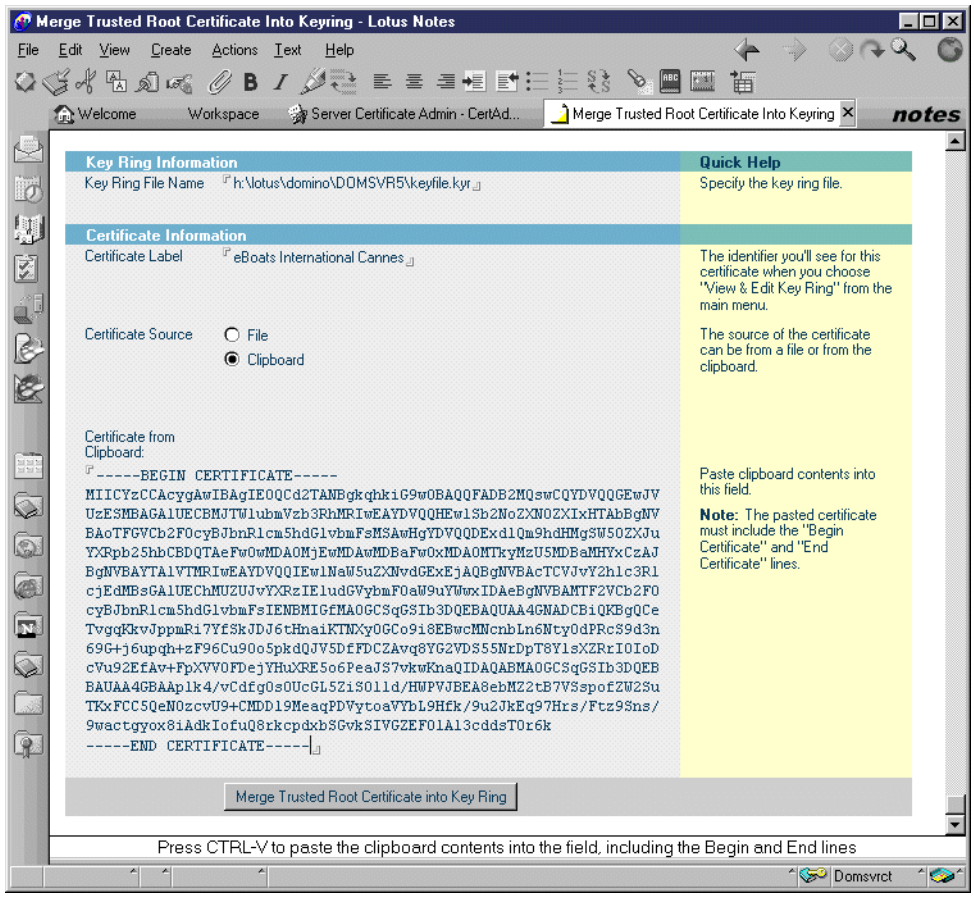


Figure 223. Install Trusted Root Certificate document ready to be submitted

10. Click **Merge Trusted Root Certificate into Key Ring**.

11. Enter the password for the server key ring file. A new display appears, as shown in Figure 224.

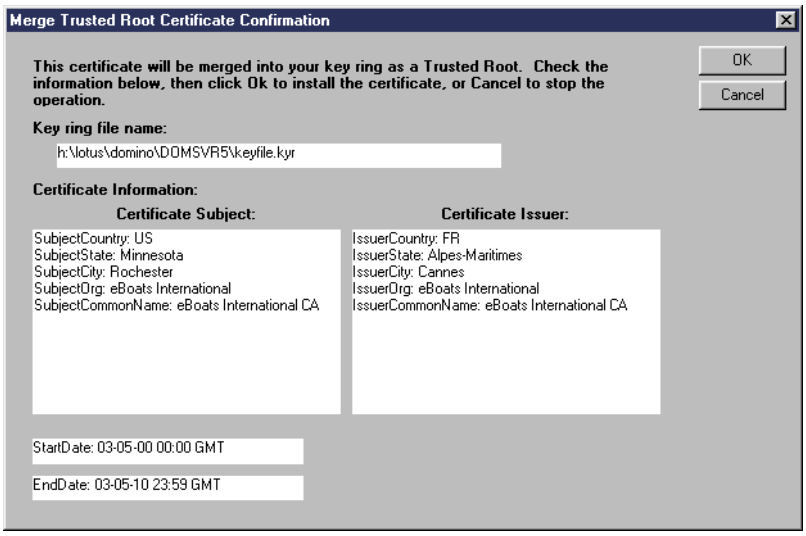


Figure 224. Merge Trusted Root Certificate confirmation

12. Click **OK**. A new message is displayed, as shown in Figure 225 on page 294.

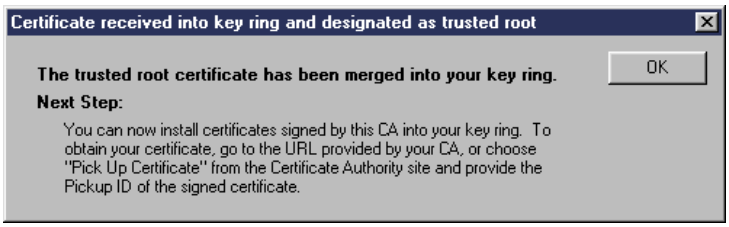


Figure 225. Merge Trusted Root Certificate Confirmation information message

13. Click **OK**. You are back in the Server Certificate Admin setup menu. See Figure 205 on page 279.

    This adds the CA's certificate as a trusted root to the server key ring file so the server trusts certificates signed by this CA.

11.1.3.4 Picking up and installing the SSL server certificate

To pick up and install the server certificate from the CA, perform the following steps:

1. Switch to your Web browser.
2. Unless you did not leave it in the meantime, go to the Web site of the CA. It is located at the URL `http://Domsvr5:8080/certca.nsf`. This takes you back to the page shown in Figure 211 on page 284.
3. Click **Pick Up Server Certificate**.
4. You found your pick-up ID in the notification received by the administrator, as shown in Figure 219 on page 289. Enter it in the prompt display shown in Figure 226.

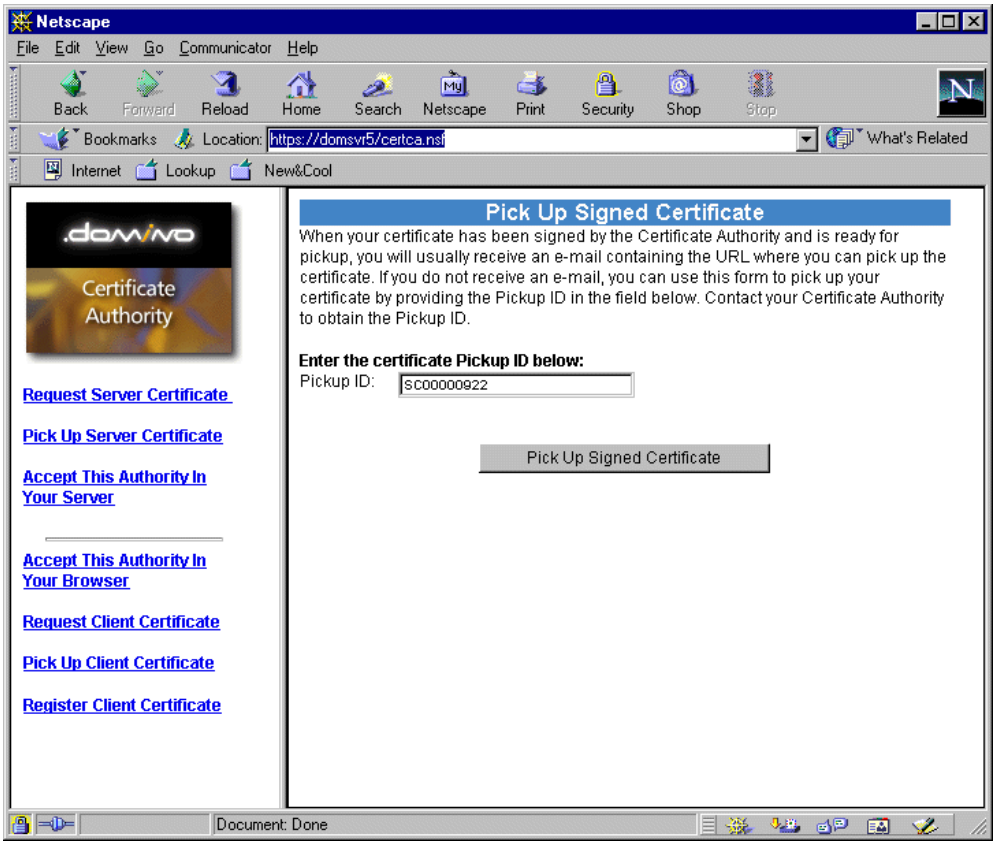


Figure 226. The pick-up ID prompt

5. Click **Pick Up Signed Certificate**.

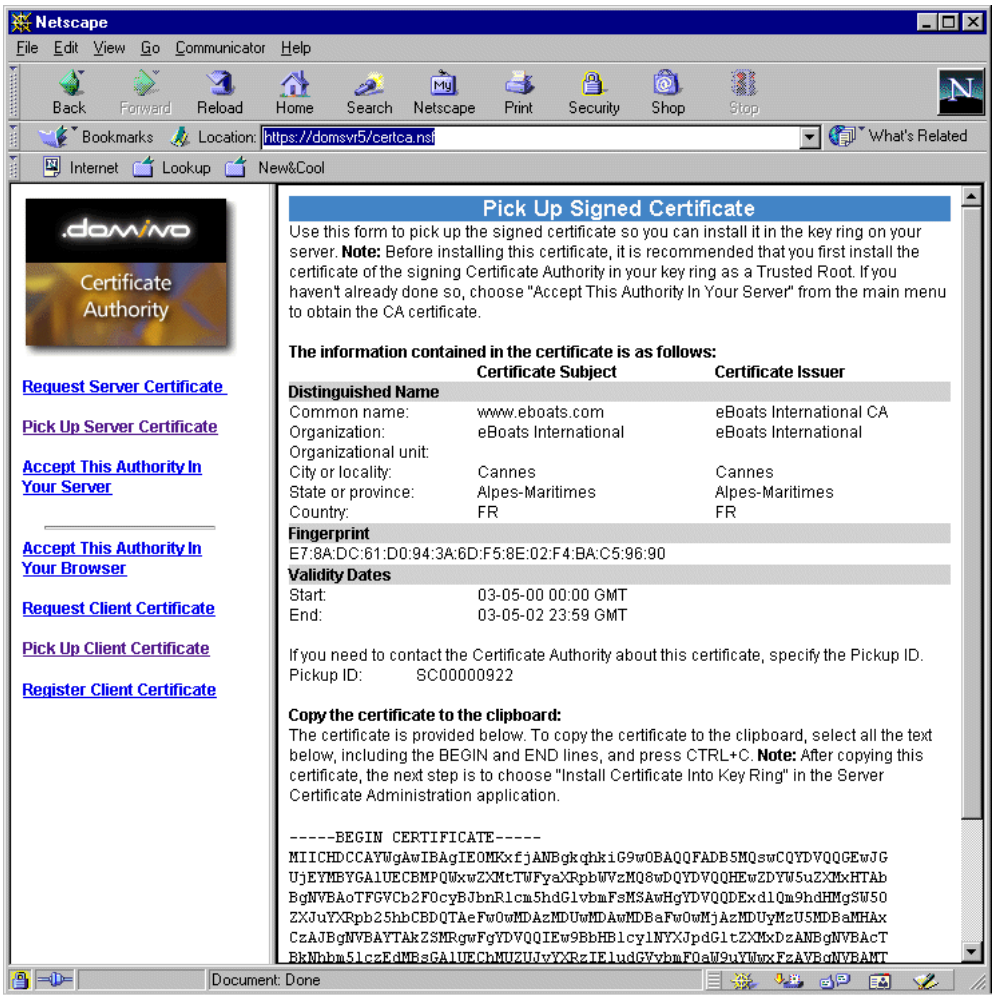


Figure 227. Pick-up Signed Certificate

6. When your certificate appears on the screen, highlight it, including the Begin and End statement lines, and copy it to the clipboard. See Figure 227 and Figure 228 for an example.

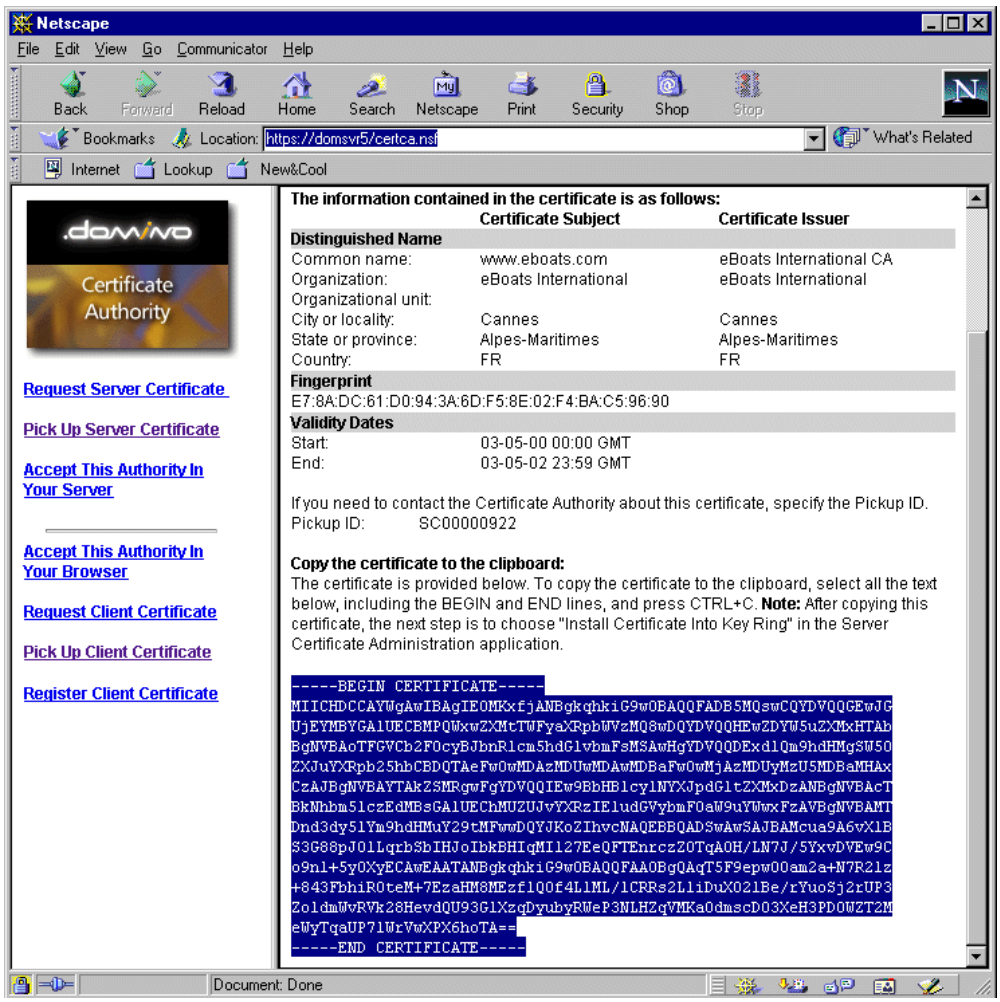


Figure 228. Copying the signed certificate

7. Switch back to your Lotus Notes client.
8. Open again the Server Certificate Admin database. The welcome panel can be seen (Figure 205 on page 279).
9. Choose the menu option 4, **Install Certificate into Key** Ring.
10. Follow the on-screen directions to enter the information that identifies the signed certificate, including the certificate source.

    The certificate source can be either a file or the copy placed in the clipboard. For eBoats, we chose Clipboard .
11. Paste the signed certificate into the Certificate from Clipboard field, as shown in Figure 229 on page 298.

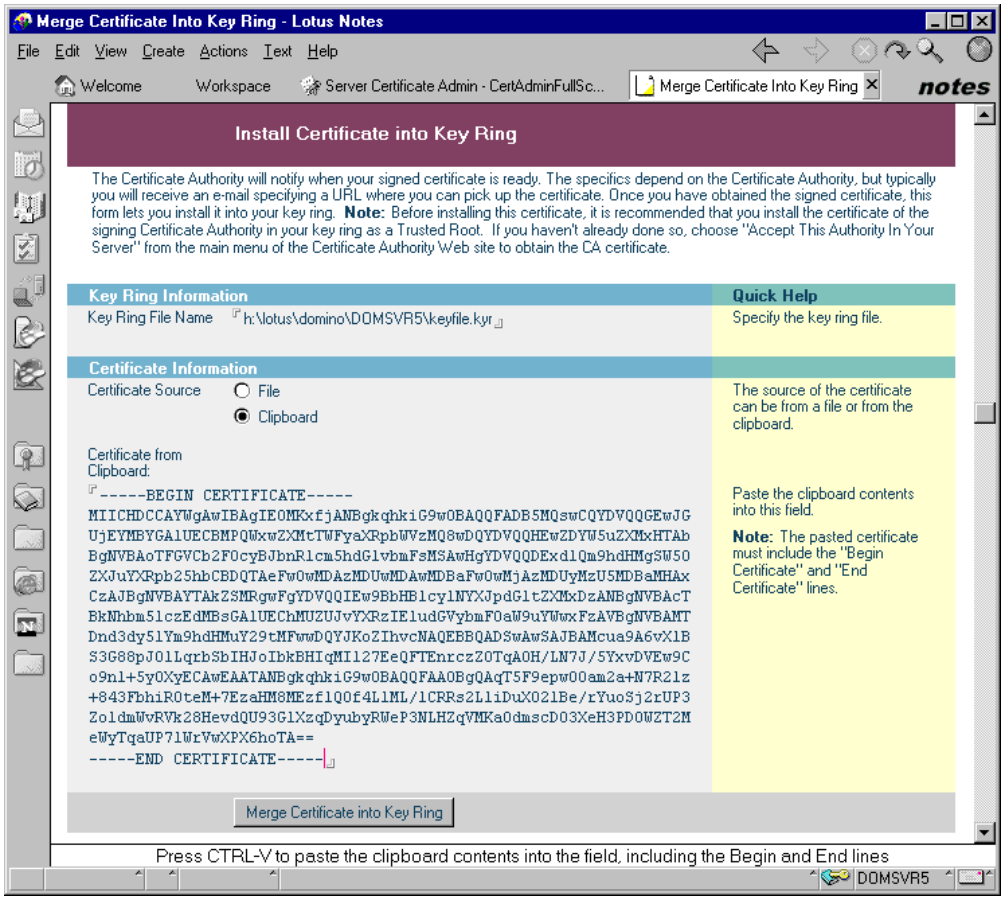


Figure 229. Install Certificate into Key Ring

12. Click the **Merge Certificate into Key Ring** button.

13. Enter the password for your server SSL key ring, and click **OK**. Figure 230 shows the confirmation that is displayed next.

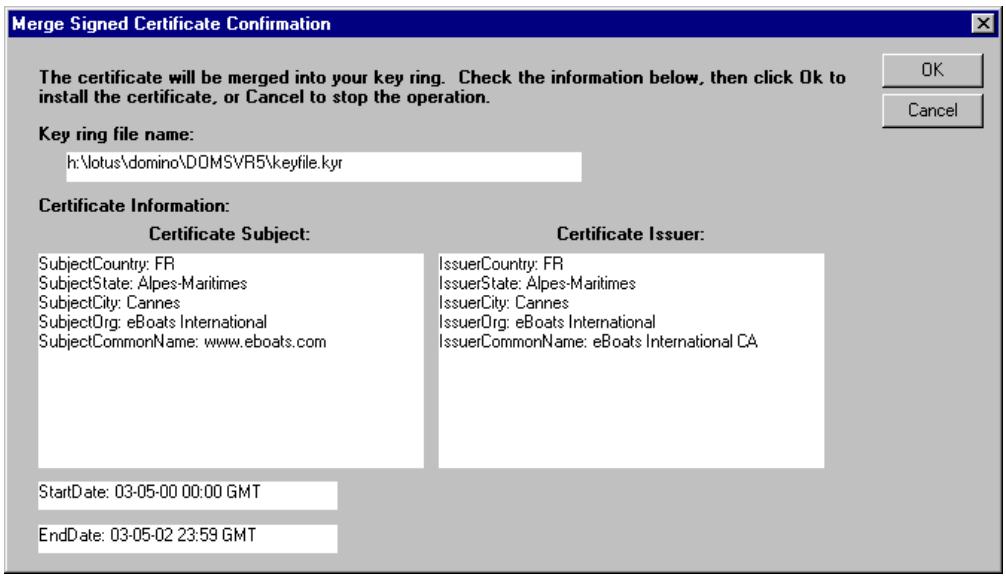


Figure 230. Merge Signed Certificate Confirmation

14.Click **OK**. Another window appears, as shown in Figure 231.

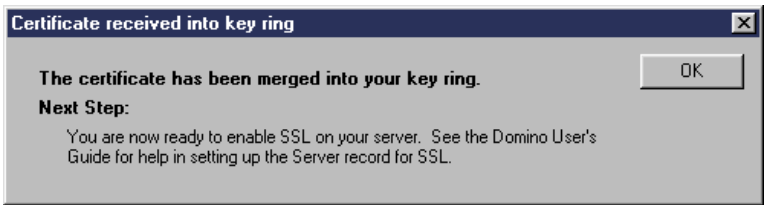


Figure 231. Merge signed certificate confirmation information message

15. Click **OK**.

    This brings you back to the Server Certificate Admin setup menu, which is shown in Figure 205 on page 279.

You are now ready to enable SSL on your server.

11.2 Configuring the port for SSL

You can configure the SSL port on a server to use one of two authentication methods:

- Server authentication only
- Server and client authentication

The sample application environment is set up for Server authentication only.

You must set authentication on a protocol-by-protocol basis. Some protocols do not support client authentication.

To set up a port for SSL, you:

- Configure the port
- Determine whether you require users to access the server using only SSL or both SSL and TCP/IP

11.2.1 Port configuration

The port configuration is divided into two parts:

- SSL settings
- Several tabs to choose the SSL settings for different protocols (Web, Directory, News, Mail, IIOP). Since we use only the Web for the sample application, Figure 232 shows the SSL settings for the Web.

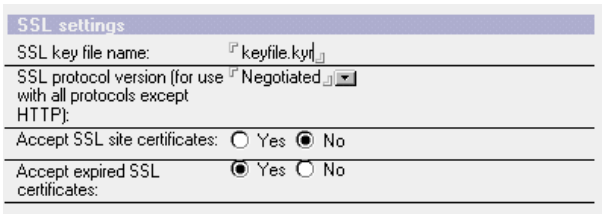


Figure 232. SSL settings

As shown in Figure 232 on page 299, the following SSL settings are implemented:

- The SSL key file is the file name of the server key ring file that the server uses.
- Accept SSL site certificates is set to No, which means the server does not accept the site certificate, and use SSL to access an Internet server, even if the server does not have a certificate in common with the Internet server.
- Accept expired SSL certificates is set to Yes, to allow clients to access the server, even if the client certificate is expired.

The SSL settings for the Web, shown in Figure 233, also have to be changed:

- Set the SSL port number to `443` (default value), which is the port number on which Domino listens for SSL requests.

  Note: If you change the default port number, clients must change their configurations as well. The default port number is usually changed only if a firewall proxy uses the reserved port number.
- Set the SSL port status to `Enabled` to allow SSL connections on the port.
- Set the Client certificate to `No`, to indicate that client authentication is not allowed.
- Set Name & password to `Yes` to use name and password authentication.
- Set Anonymous to `No`. Anonymous access is not authorized.

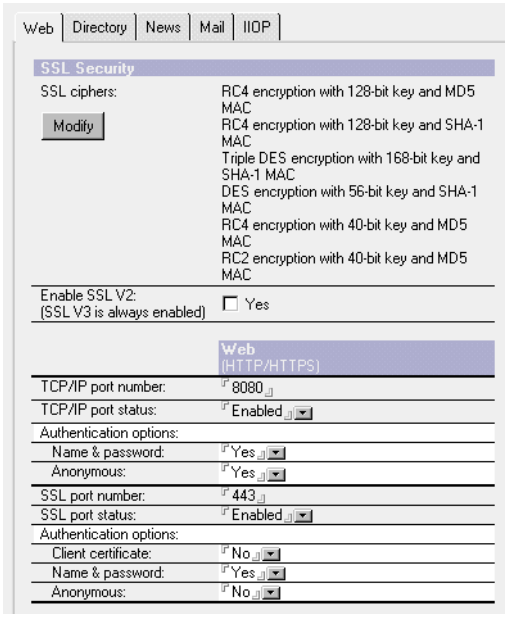


Figure 233. SSL settings for the Web

11.2.2 SSL connection request on a protocol-by-protocol basis

After you set up SSL on a server, you can require an SSL connection on a protocol-by-protocol basis. For example, you can force clients and servers to use SSL to connect to a specific server port and deny access to clients and servers that use TCP/IP to connect to that port. Require SSL connections when you want to make sure that clients use a secure connection to access the databases on the

server. If you do not require an SSL connection, clients can use either SSL or TCP/IP to connect to the server.

You can require SSL connections for all databases on a server or for an individual database. Section 11.3, “Enabling SSL at a database level” describes how to enable SSL at a database level.

11.2.3 Checking that SSL is enabled

After having stopped, and restarted your Domino server HTTP task, you can use the `tell http show security` command to check if SSL is enabled. This command is explained in 9.1.6, “Other useful HTTP console commands” on page 233.

These are the messages you should get if SSL is well implemented:

```
03/21/2000 03:01:52 PM  Base server:
03/21/2000 03:01:52 PM     SSL enabled
03/21/2000 03:01:52 PM     Key file name: /lotus/domino/domsvr5/keyfile.kyr
03/21/2000 03:01:52 PM     Secure server started
```

These are the messages on a server where SSL has not yet been fully configured:

```
04/21/2000 03:09:09 PM  Base server:
04/21/2000 03:09:09 PM     SSL NOT enabled
04/21/2000 03:09:09 PM     Key file name: /DOMINO/domsvr99/CASvrkey.kyr
04/21/2000 03:09:09 PM     Secure server not started. Waiting for HTTPS
```

Another way to check is to use the `show tasks` (or `sh ta`) command, which lists all the Domino server active tasks. You should find the entry:

```
HTTP Web Server      Listening on port(s) 8080, 443
```

11.3 Enabling SSL at a database level

To protect transactions in the eboatsB2B.nsf and eboatsIntranet.nsf databases, and force SSL usage, the **Web Access: Require SSL Connection** property must be selected.

To define the Web Access: Require SSL Connection database property, select or open the database, and choose **File->Database->Properties**. The property can be found under the first tab from the left.

The setting for the eboatsB2B.nsf database is shown in Figure 234 on page 302.

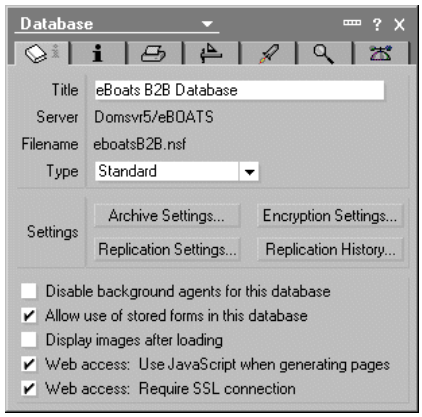


Figure 234. Web Access: Require SSL Connection database property

The same setting is used for the eboatsIntranet.nsf database.

Note

You need to review all other Domino databases on the Domino server for which this property is necessary.

You are now ready to secure connections to the eBoats International Web site, in regards to the Domino server.

11.4 Enabling SSL at a servlet level

There is no external way to force SSL usage for servlets. As explained in 10.4.1.1, “Protecting servlets” on page 258, there is no use in accessing directly the two servlets available in the sample application. However, if you want to be sure that SSL is used, you can change the servlet code to check that SSL is used and to force a redirection to another URL if the URL does not start with “https”.

The Java code for this is:

```
if (req.getScheme().equalsIgnoreCase("https")) {
   // Process request
}
else {
   String sslUrl = "https://" + req.getServerName() + req.getRequestURI();
        if (req.getQueryString() != null)
        sslUrl += "?" + req.getQueryString();
   response.sendRedirect(sslUrl);}
```

The *sendRedirect* method, as described in the JSDK documentation, sends a temporary redirect response to the client using the specified redirect location URL. The URL must be absolute (for example, `https://hostname/path/file.html`). Relative URLs are not permitted here.

The code of the sample application’s servlets does not include checking that SSL is used to start the servlet. However, we tested this solution, to force SSL usage, in a separate Java servlet. The code for this servlet can be found in C.5, “Servlet forcing SSL usage” on page 370.

11.5 Additional considerations

Using SSL digital certificates for securing the connection between the clients and the server require that port 443 on the server is activated, as described in 11.2, “Configuring the port for SSL” on page 299. By activating port 443 and server authentication, we need to configure the firewall to accept request for this port.

This is a requirement for any scenario involving the firewall to protect our public Web server regardless of whether Domino is running on a separate system from the DB2 system or Domino and DB2 are running on the same system. The essential point is whether the firewall is protecting our public Web server.

Therefore, the firewall has to permit traffic accessing the port 443 (HTTPS) on the public Web server as well as port 80 (HTTP). The difference between the two are secure and non-secure connections.

When using the scenario where Domino is running on a separate system and we want to access the system running DB2 UDB for AS/400 through the firewall, we need to permit DRDA traffic between the public Web server on the DMZ and the backend system on the internal network. DRDA is using port 446.

We do not need this support in the firewall if we are using the scenario where Domino and DB2 UDB for AS/400 are running on the same AS/400 system behind the firewall. Instead, we need to activate the Network Address Translation in the firewall to map a public IP address with the internal private IP address used for interface running the Web server. This will make the Web server look public from the Internet despite being on the internal secure network.

This also gives us the flexibility of changing the internal private IP address for the Web server if necessary without changing the public definitions for the Web server. However, we need to redefine the mapping directives in the firewall in this case.

11.6 Saving your Domino server configuration and environment

After you successfully complete the steps required in this chapter and in the preceding two chapters, *don’t forget to save your Domino server environment*.

Domino for AS/400 takes advantage of the AS/400 single-level store architecture. Domino databases and programs are spread across all the AS/400 disk units, along with other AS/400 objects. The OS/400 operating system automatically manages the allocation of disk space so that you do not have to. To back up information on the AS/400 system, you back up *logically* (by library or directory), not physically (by disk unit). To plan a backup strategy, you need to understand the logical location of your Domino for AS/400 databases and programs.

To protect the data from disasters (such as a site loss or hardware loss) and from human error, such as accidentally deleting a critical database, *develop a good strategy for regularly backing up* the information on your Domino server. Make a plan to back up:

- Objects that change infrequently, such as programs for the Domino product. We recommend that you to save the product before any fix (Quarterly

Maintenance Update, hotfix) or new release (Quarterly Maintenance Update) installation.

- Objects that change depending on the configuration needs, such as the NOTES.INI file, or the file.
- Objects that change regularly, such as Domino databases, including the Domino Directory.

The steps to save your environment are described in “Managing Backup and Recovery”, in the Domino for AS/400 Help database. This document describes how to implement backup and recovery for a Domino server running on the AS/400 system. You use AS/400 commands to perform the backup and recovery tasks.

The Domino for AS/400 Help (as400hlp.nsf) database contains all Domino AS/400-specific information. It is also delivered with the software as the printed book, *Installing and Managing Domino for AS/400*.

Note

The sample application also needs the DB2 UDB for AS/400 collection to be saved, as well as the operational data (customer data, business partner data, orders) reside in DB2 UDB for AS/400. Both backup operations should be coordinated.

Chapter 12. Using the sample project application

This chapter, describes how to use the sample applications developed for our fictional company eBoats International. These databases were designed to be accessed from both a browser and a Lotus Notes client.

Lotus Domino is ideally suited for Internet, extranet, and intranet environments. The sample applications demonstrate the use of Domino in all of these environments. Step-by-step instructions on using the sample applications are divided into the following three sections:

- Functions open to the public users (Internet)
- Functions available only to authorized Business Partners (extranet)
- Functions available only to eBoats International employees (intranet)

The following sections describe in detail the features available within each of the sample databases.

12.1 Functions open to public users (Internet)

Any user who accesses eBoats International Web site is treated as a public user. Public users are not required to have a user ID and password to access the database. eBoats International's Web site is contained in the eboatsInternet.nsf database. From this database, users can view information on the entire product line, request more information, locate the nearest business partner, and submit an application to become a business partner. Figure 235 shows what a public user sees when they access eBoats International's Web site.

Figure 235. eBoats International's Web site

The following sections describe in detail each menu option available to public users.

12.1.1 Product Info

To view information on eBoats International's product line, a user clicks on Product Info from the menu. They are presented with a view listing every product line and model. This view is shown in Figure 236.

Figure 236. List of product lines and models

To view detailed information on a particular product, the user clicks on the appropriate product line option. A list of products is shown. The user clicks on the model in which they are interested. A display containing detailed information on the selected product is shown. An example of the information displayed is shown in Figure 237.

Figure 237. Detailed product information

12.1.2 Request Info

If a user wants to obtain more information on a product, where to purchase products, or the company, they submit an information request form online. To display the form, the user clicks Request Info from the menu. The form is shown in Figure 238 on page 308.

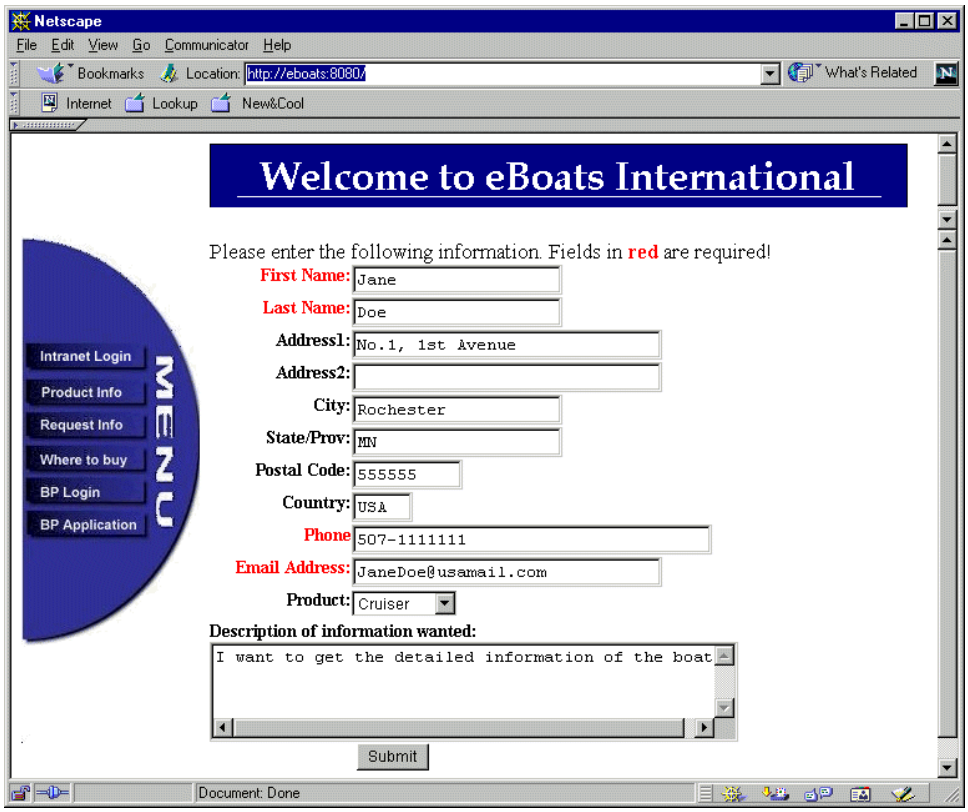


Figure 238. Information request form

The user is required to fill in all required fields which are indicated in red on the display. Once they are finished, they click the Submit button. A confirmation of their request is displayed. To return to the welcome page, they click the Home Page link shown in Figure 239.

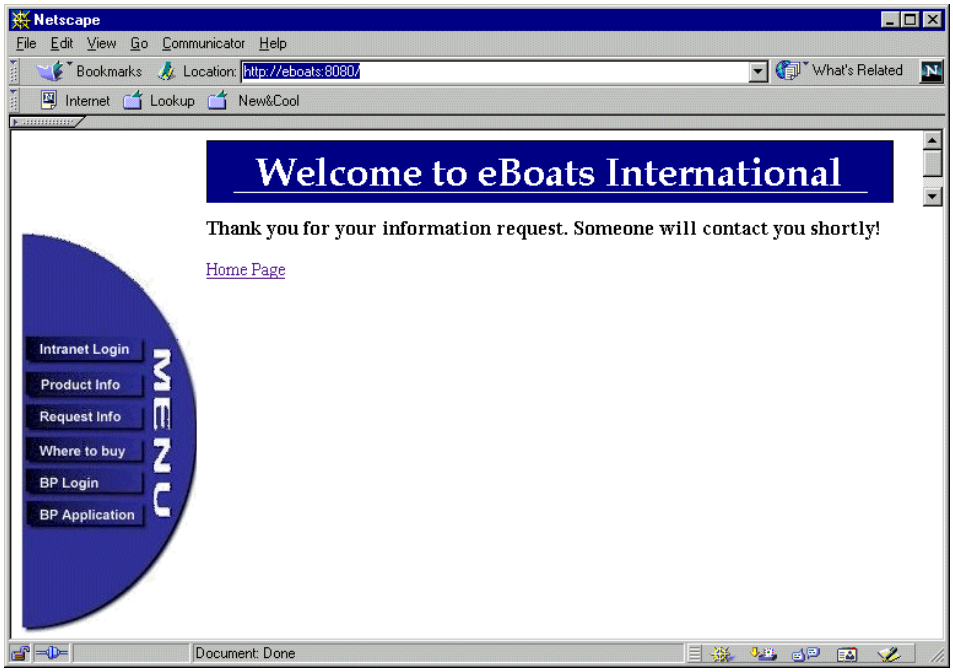


Figure 239. Confirmation of information request

12.1.3 Where to buy

eBoats International only sells its products through authorized business partners worldwide. Users wanting to purchase products, can find the nearest business partner by clicking on Where to buy in the menu. A world map is displayed as shown Figure 240.

Figure 240. World map

Each region of the map has a hotspot that a user can click to view all business partners in that region. When a user clicks on a region, a view showing all business partners in that region is displayed, as shown in Figure 241.

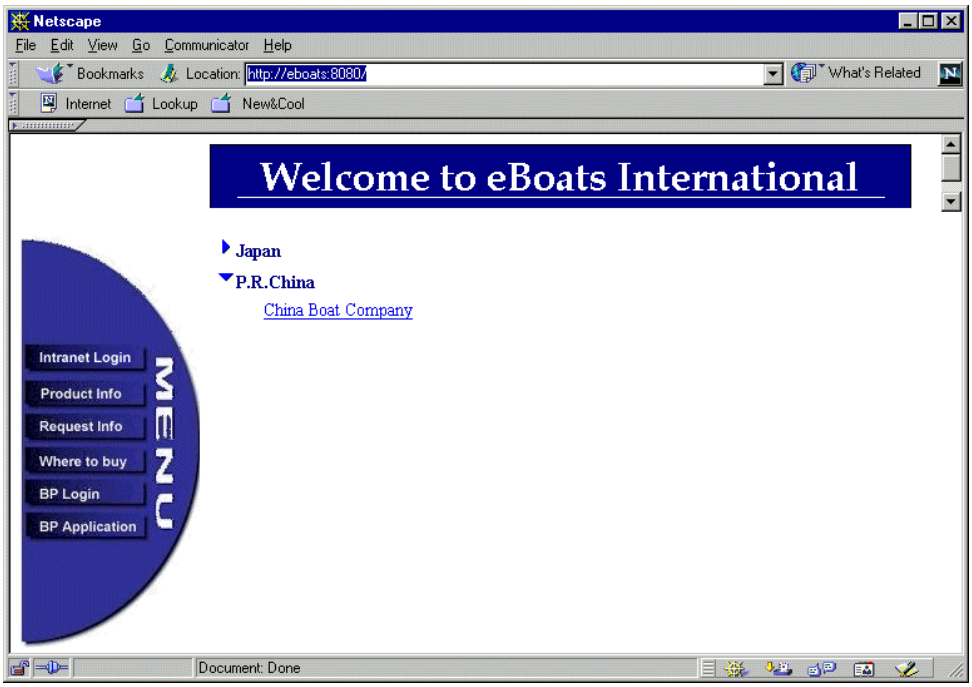


Figure 241. List of business partners

For more detailed information on a particular business partner, click the business partner's name. An example of the detailed information displayed is shown in Figure 242.

Figure 242. Detailed business partner information

12.1.4 BP Application

Companies wanting to sell eBoats International's product line need to submit an application. This application can be submitted online by clicking on BP Application from the menu. The application form is shown in Figure 243.

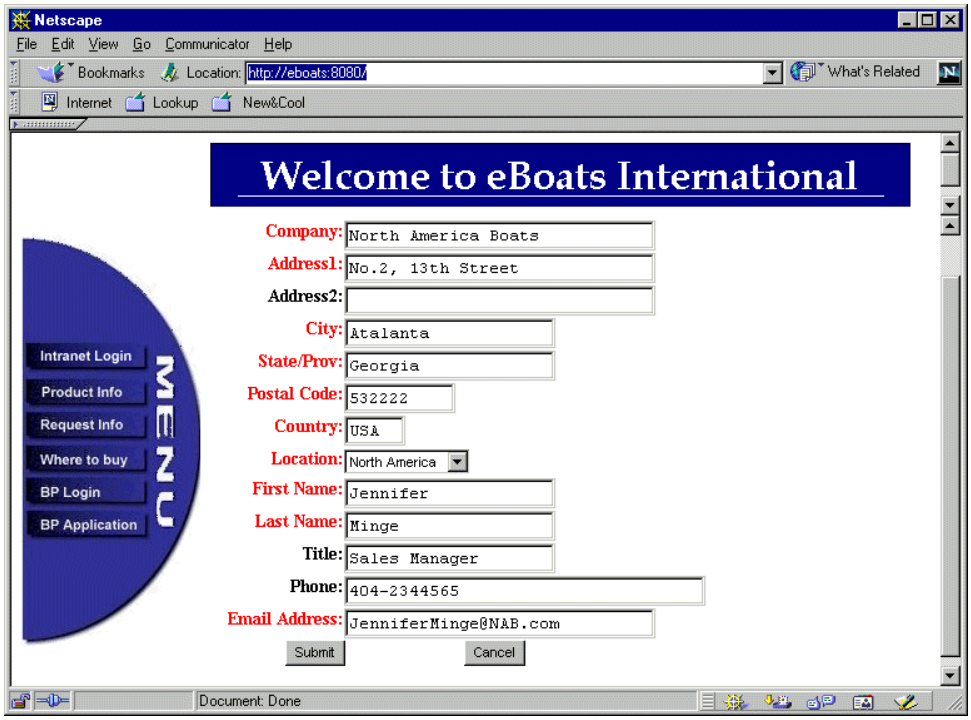


Figure 243. Business partner application form

The user is required to fill in all required fields which are indicated in red. Once they are finished, they click the Submit button. A confirmation of the application is displayed. To return to the welcome page, the user clicks the Home Page link shown in Figure 244 on page 312.

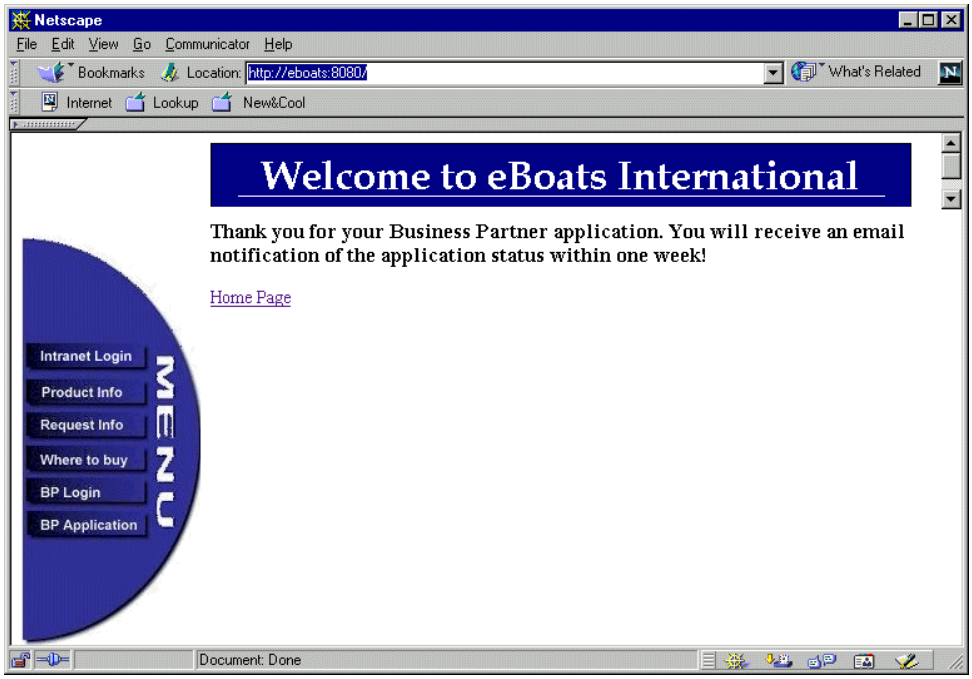


Figure 244. Confirmation of the business partner application

12.1.4.1 Approve Business Partner

In the eboatsBP.nsf database, there is a profile document that contains the names of all employees who should be notified when an application is submitted. These people are notified through e-mail. If the user is listed as an approver, they will receive an e-mail similar to the one shown in Figure 245.

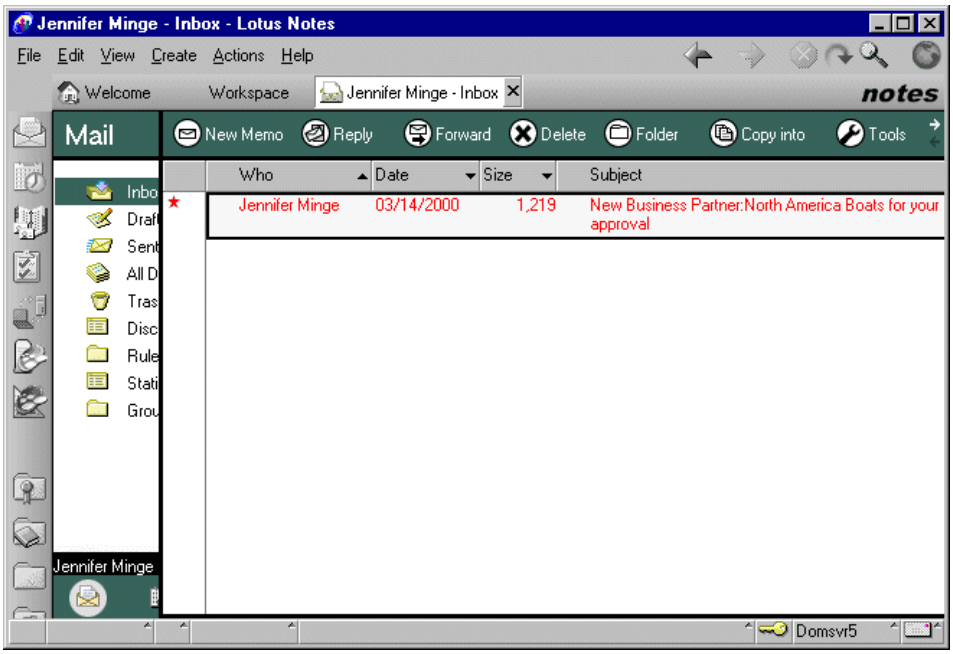


Figure 245. New business partner notification e-mail

Then, the user double-clicks the message to display the contents. The message provides instructions on how to review the application and a doclink to the application. The memo is shown in Figure 246.

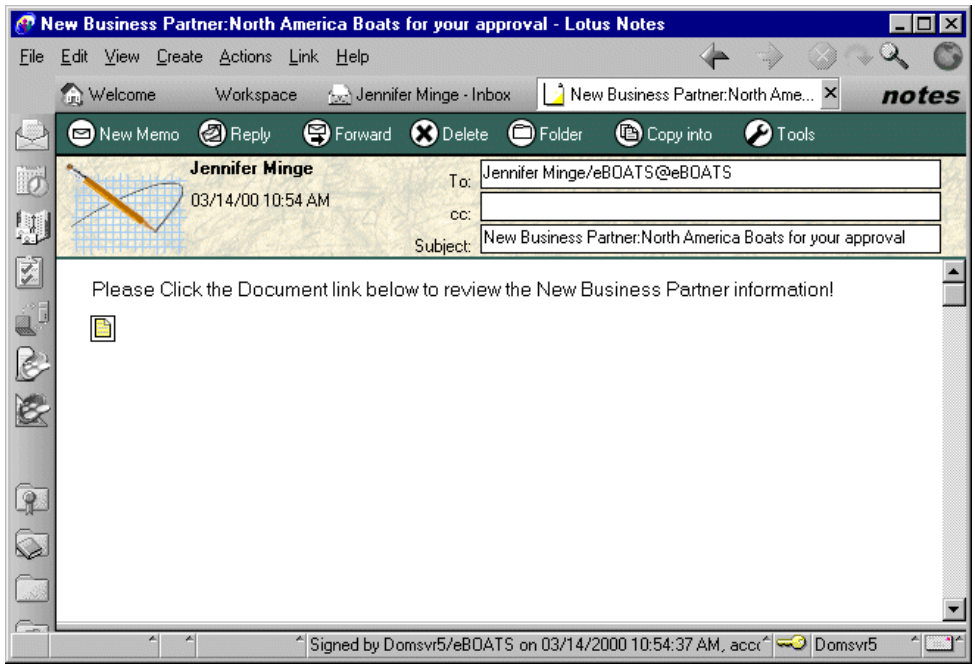


Figure 246. The contents of the notification message

The user clicks the link below the text to open the approver form, as shown in Figure 247.

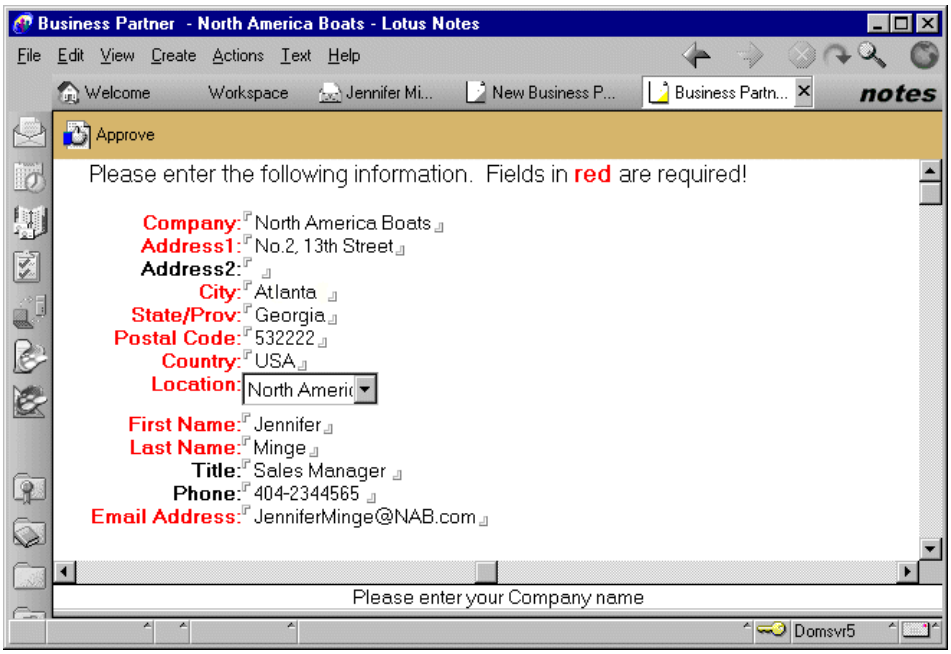


Figure 247. Approvers form

The user clicks the Approve button to approve the new business partner application. A new business partner record is created into the BPINFO DB2 UDB

for AS/400 table. The business partner number and password are sent to the applicant through e-mail.

12.2 Functions available for the company's business partners (extranet)

Authorized business partners access the eBoats International Web site just like all public users. From the Web site, there is a menu option for the business partners to logon to a restricted extranet database. This database is eboatsB2b.nsf.

When a business partner clicks BP Login, they are prompted to enter their user ID and password. They use the user ID and password that were assigned when their application to become a business partner was approved. This logon display is shown in Figure 248.

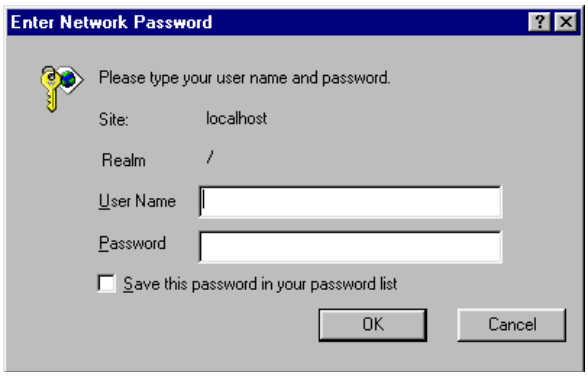


Figure 248. Business partner logon display

Once a business partner logs on, the display shown in Figure 249 appears.

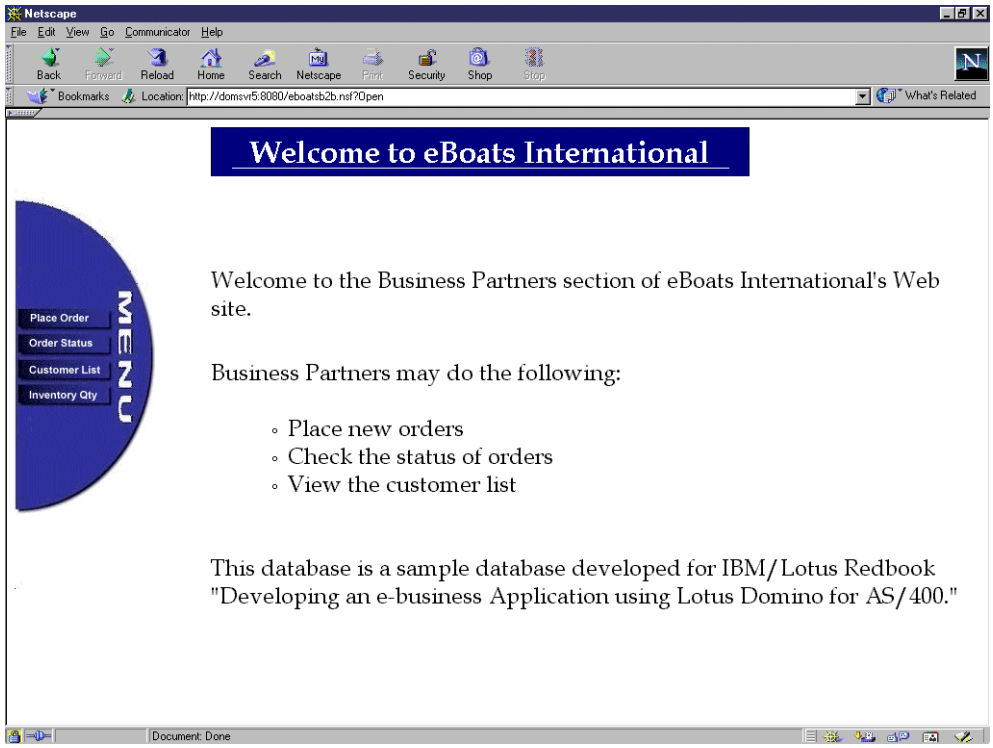


Figure 249. Business partner main display

From this database, business partners can place orders, check order status, view the customer list, and check inventory levels. Each of these menu options are described in the following sections.

12.2.1 Place Order

When business partners want to place a new order, they click Place Order from the menu. The shopping basket form is displayed as shown in Figure 250 on page 316.

Figure 250. Shopping basket form

A list of all models with their model numbers and price is shown in the drop-down box. To place an order, a user selects the appropriate product from the list, enters the quantity, and clicks the Order Now button. A user can order as many products as they want. After ordering all products, they click the Place Order button to submit the order to eBoats International. The order is entered into the ORDERINFO DB2 UDB for AS/400 table, an order number is generated for each product, and the display shown in Figure 251 is presented as confirmation that the order was successfully processed.

Figure 251. Order confirmation display

This confirmation display contains a separate order number for each product ordered. The user needs to remember this order number, if they want to check the status of their orders later.

12.2.2 Order Status

Business partners can check the status of their orders at any time by clicking on Order Status from the menu. A cookie containing the business partner number is created when a business partner logs on to the extranet database. When a user clicks Order Status, a Java agent reads the cookie to get the business partner number, queries the ORDERINFO DB2 UDB for AS/400 table, and returns all orders for this business partner. An example of the information displayed is shown in Figure 252 on page 318.

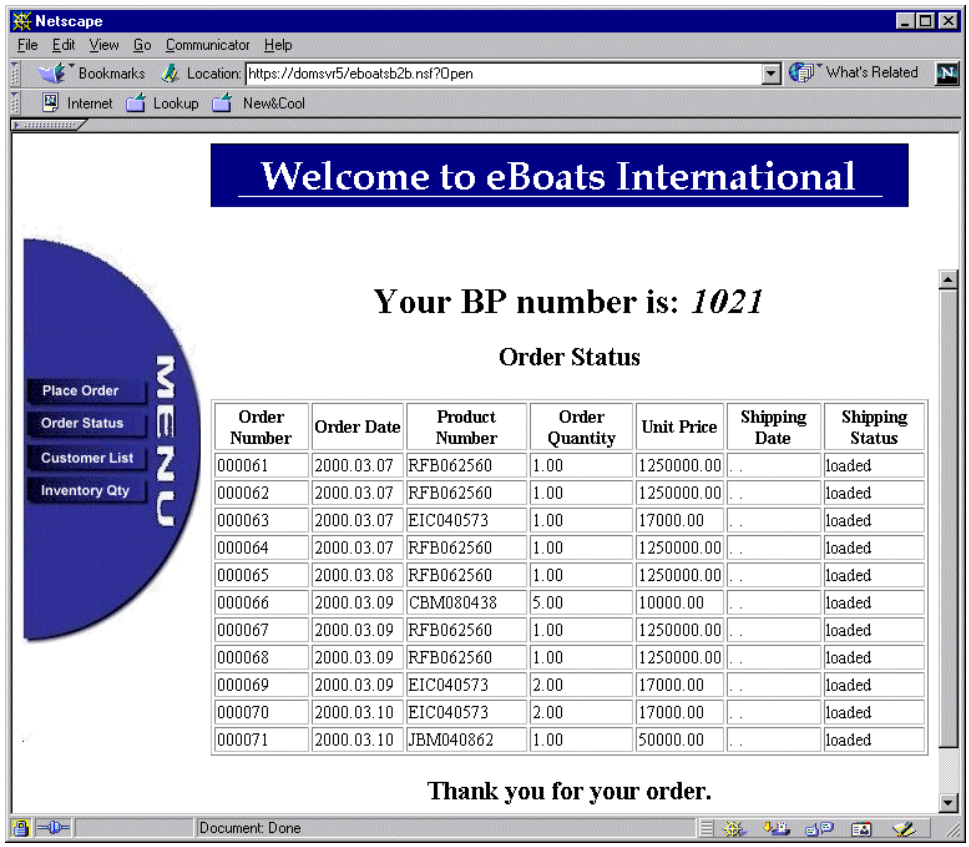


Figure 252. Order status display

12.2.3 Customer List

A new record is created in the CUSTOMER DB2 UDB for AS/400 table every time a user submits a request for information form online. Business partners can view a list of these users at any time by clicking Customer List from the menu. A view showing all customers organized by country is displayed, as shown in Figure 253.

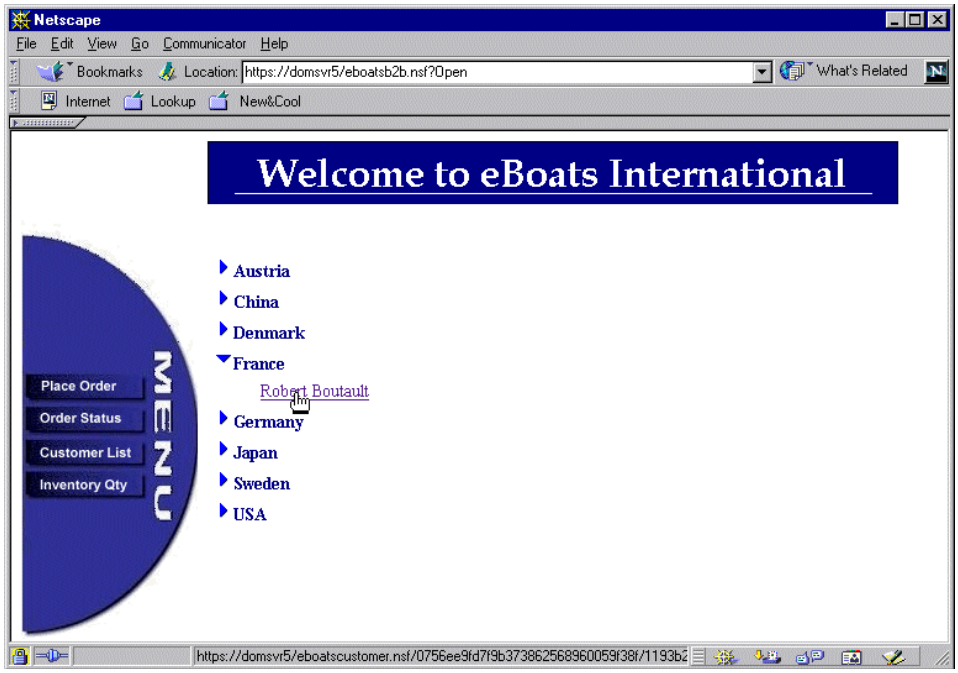


Figure 253. Customer list

For more detailed information on a particular customer, a business partner clicks the customer's name. An example of the detailed information displayed is shown in Figure 254.

Figure 254. Customer information

12.2.4 Inventory Qty

If a business partner wants to know the current inventory level for any product, they click Inventory Qty from the menu. The display that appears demonstrates

three different methods of obtaining the current inventory levels from the PRODUCTS DB2 UDB for AS/400 table. The user clicks on the appropriate model from the drop-down box and then clicks any one of the three buttons. The current inventory level for the selected product is displayed in the inventory level field. The display to check inventory quantity is shown in Figure 255.

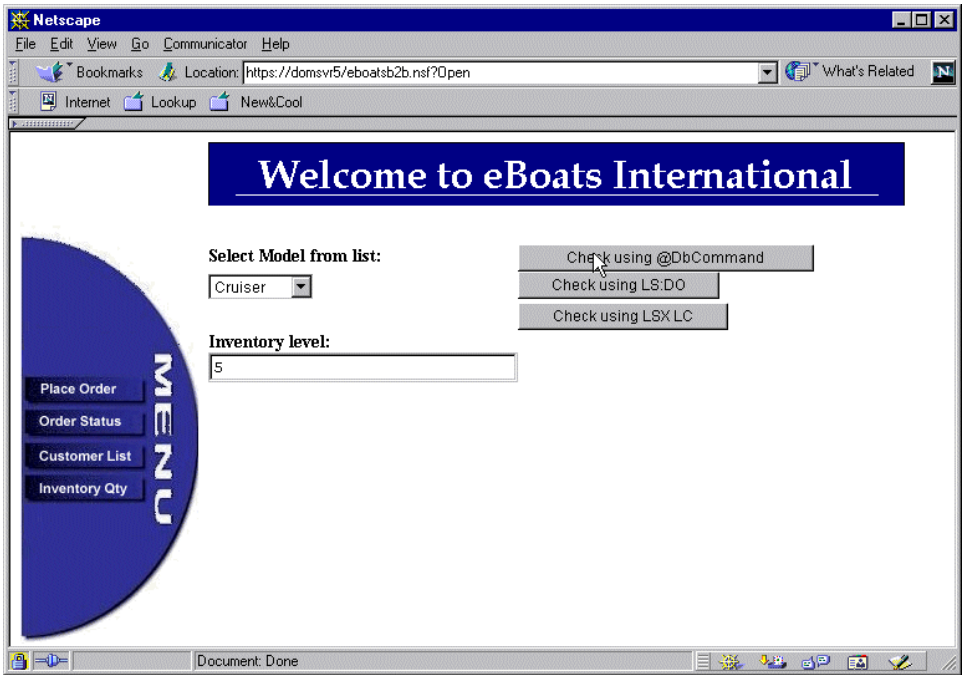


Figure 255. Inventory quantity

12.3 Functions available inside the company (intranet)

eBoats International employees access the Web site just like all public users. From the Web site. There is a menu option for employees to logon to a restricted intranet database. This database is eboatsIntranet.nsf.

When an employee clicks Intranet Login, they are prompted to enter their user ID and password. They use their user ID and Internet password that is stored in their person document in the Domino Directory. This logon display is shown in Figure 248 on page 314.

eBoats International employees use this database to track the status of business partner's orders. Employees have the option of viewing all orders or just the orders for a selected business partner. Figure 256 shows the main display for this database.

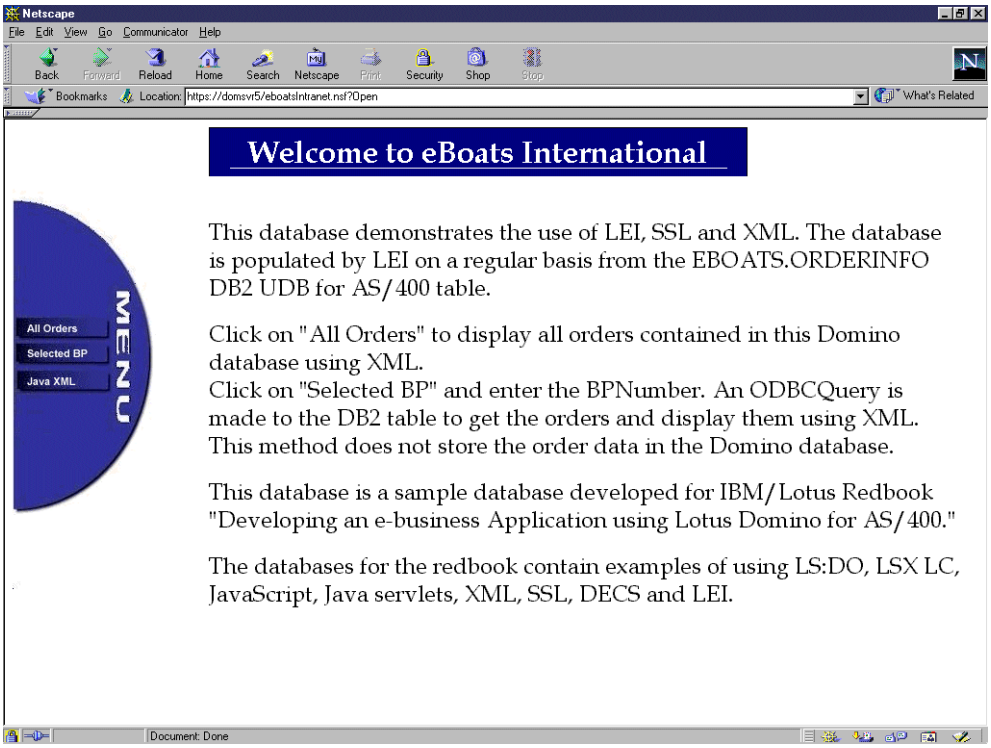


Figure 256. Intranet database home page

The following sections describe the menu options available to eBoats International employees in this database.

Attention

This information is displayed using XML, which requires Internet Explorer 5 or higher. Internet Explorer can apply XSL style sheets that produce HTML, allowing direct browsing of the XML files.

12.3.1 All orders

Users can display the current status of all orders contained in this Domino database. This information may be accurate. When business partners place orders, a new record is created in the ORDERINFO DB2 UDB for AS/400 table. This table is replicated to the Domino intranet database every hour, using an LEI direct transfer activity. With replication occurring on an hourly basis, employees may not be shown all the orders contained in the DB2 UDB for AS/400 table.

Figure 257 on page 322 shows the XML display presented when an employee clicks All Orders.

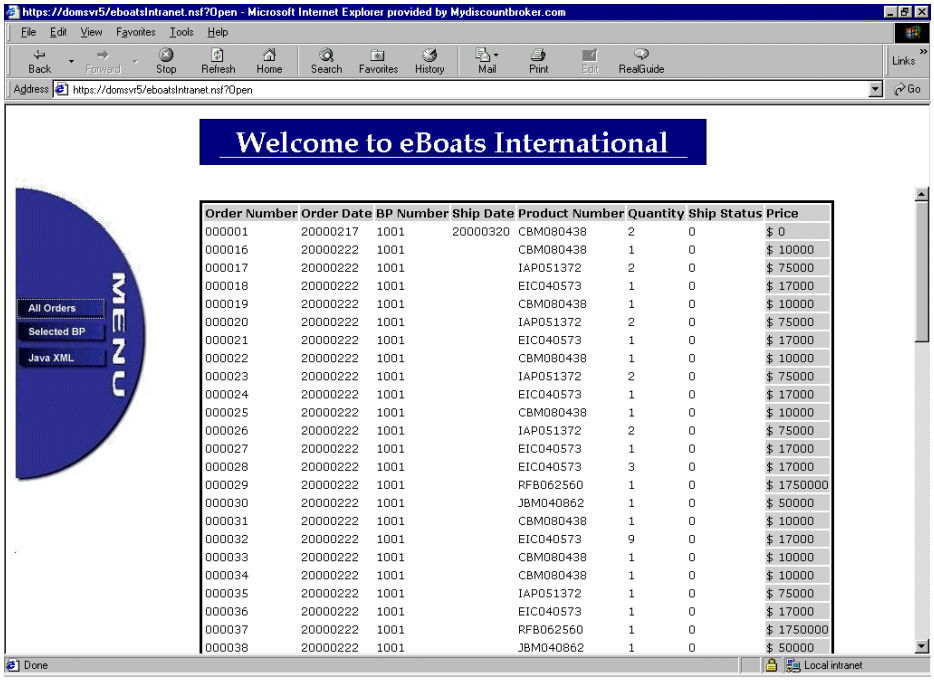


Figure 257. XML display of all orders

12.3.2 Selected BP

Employees have the option of displaying only the orders for a selected business partner. This allows employees to track order status for business partners in their region. When an employee clicks Selected BP, they are presented with the display shown in Figure 258 prompting them to select the desired business partner number.

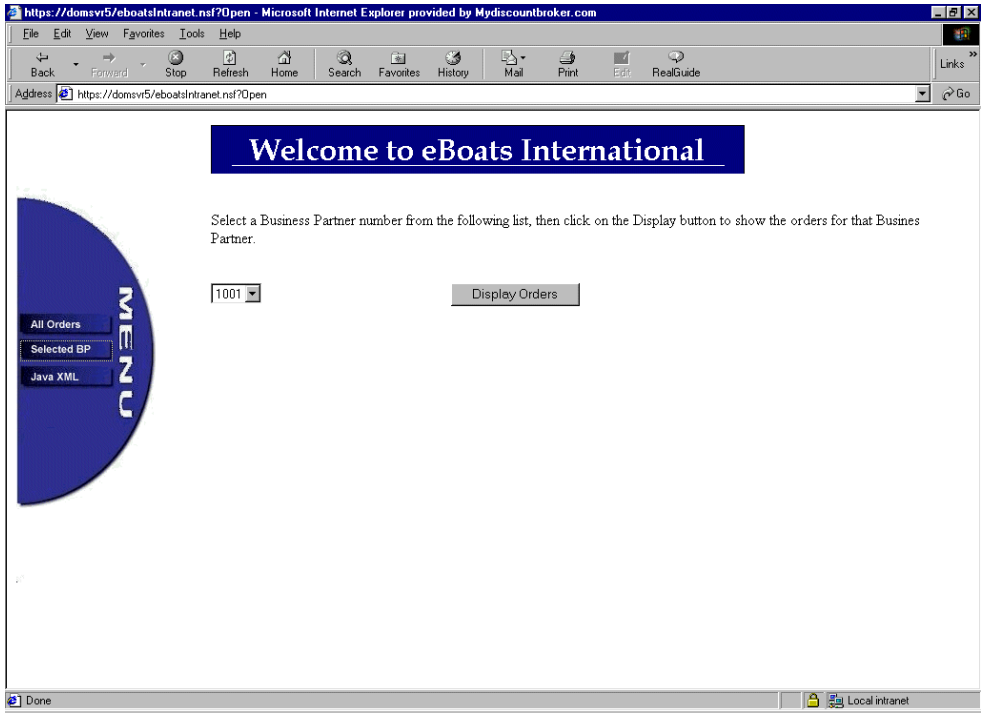


Figure 258. Screen prompting for selected Business Partner number

The employee selects the business partner number from the drop-down box and then clicks the Display Orders button. Figure 259 shows the output for the specified business partner.

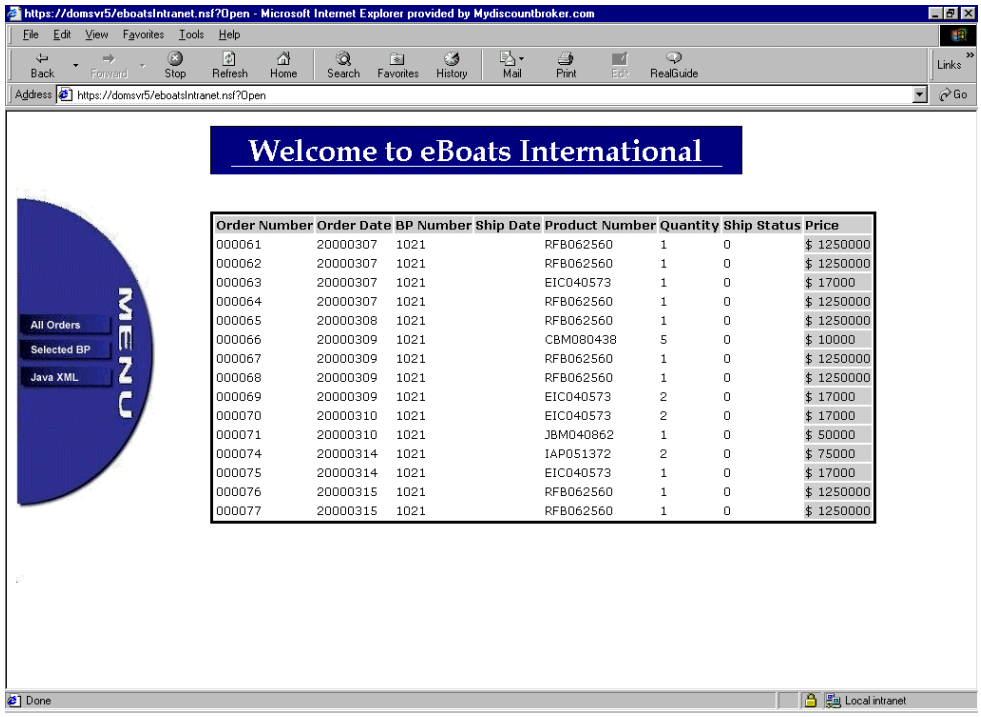


Figure 259. Orders for a selected business partner

12.3.3 Java XML

This option provides the same functionality as All Orders in that it shows the status of all orders. The difference is this option calls a Java agent that directly queries the ORDERINFO DB2 UDB for AS/400 table. The information is displayed in real-time because it accesses the DB2 table directly.

The All Orders option may not always show the same information, because the displayed data is the data copied from DB2 UDB for AS/400 during the last run of the replication activity (described in 8.2.3, “Intranet database: eboatsIntranet.nsf” on page 196). In the meantime, additional orders may have been entered.

When a user clicks Java XML, a display appears showing all orders contained in the ORDERINFO DB2 UDB for AS/400 table. Figure 260 shows this display.

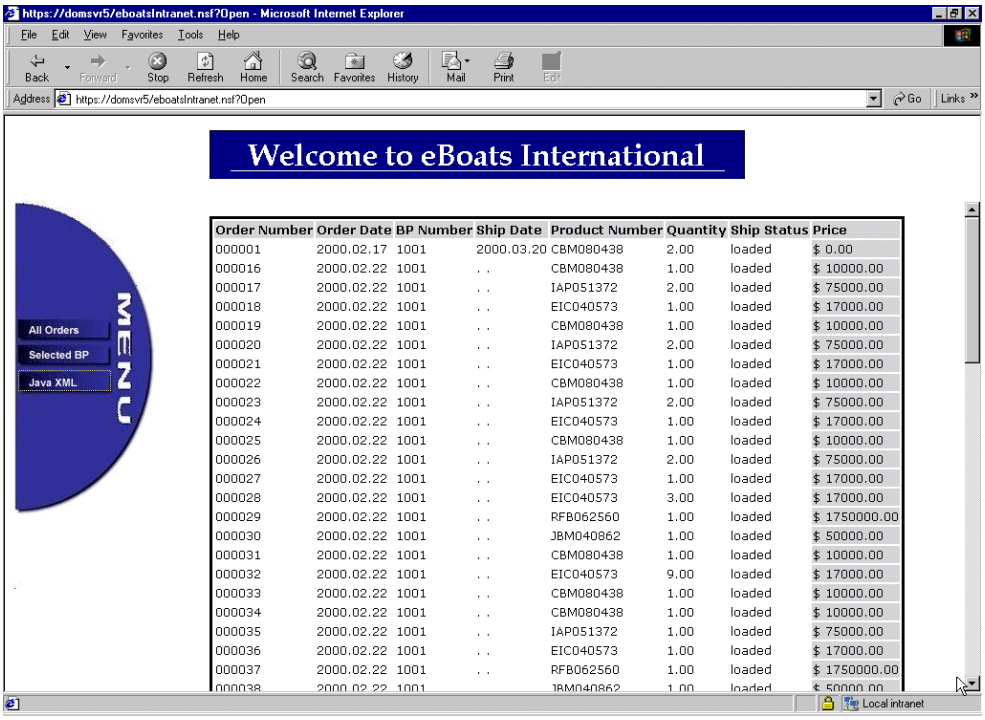


Figure 260. Java XML display

Part 3. Appendices

Part 3 contains several appendices, covering the following topics:

- LotusScript sample code
- JavaScript sample code
- Java sample code
- Other Java sample code (@DB Command, SQL stored procedure, XSL)
- Domino integration with WebSphere on the AS/400 system
- Domino data access control

Appendix A. LotusScript sample code

This appendix lists all the LotusScript code examples presented in the sample databases. Each section is labeled with the name of the function or agent in which the code is contained.

A.1 NewBPNotification agent

This agent is contained in the eboatsBP.nsf database. The agent is called by the WebQuerySave agent on the the BP form in this database. This agent notifies the appropriate eBoats International employees when a business partner application is submitted online.

- **Options**

```
Option Public
```

- **Declarations**

```
Dim session As NotesSession
Dim db As Notesdatabase
Dim newDoc As NotesDocument
Dim rtitem As NotesRichTextItem
Dim newBPDoc As NotesDocument
Dim newBPName As String
Dim view As NotesView
Dim appDoc As NotesDocument
Dim appStatus As String
```

- **Initialize**

```
    On Error Goto errorHandler

    Dim agentLog As New Noteslog("agentLog")
    agentLog.OpenAgentLog

    agentLog.LogAction( "agent newBPNotification")

    Set session = New  NotesSession
    Set db = session.CurrentDatabase
    agentLog.LogAction( "got db")

    'Save the current Business Partner Document
    Set newBPDoc = session.DocumentContext
    appStatus = newBPDoc.GetItemValue("Status")(0)
    agentLog.LogAction("got status")

    If ( appStatus <> "0" ) Then
        Exit Sub
    End If
    newBPDoc.Status = "1"

    newBPName = newBPDoc.GetItemValue( "Company" )(0)

    'Create a new memo
    agentLog.LogAction( "start a new memo")

    Set newDoc = New NotesDocument(db)
    Set rtitem = New NotesRichTextItem( newDoc, "Body")
    Call rtitem.AppendText("Please Click the Document link below to review the New
Business Partner information!")
    Call rtitem.AddNewLine(2)
    Call rtitem.AppendDocLink(newBPDoc, "New Business Partner for approval")

    newDoc.Subject = "New Business Partner:" & newBPName & " for your approval"

    Set view = db.GetView( "Approvers" )
    Set appDoc = view.GetFirstDocument
    newDoc.SendTo = appDoc.ApproverValues
    newDoc.Form = "Memo"

    Call newDoc.Send(True)
```

```
    agentLog.LogAction(  "memo sent")
    agentLog.Close
    Exit Sub

errorHandler:
    Print Err() & " Error : " & Error$
```

A.2 RegisterBP agent

This agent is contained in the eboatsBP.nsf database. It uses LS:DO. This agent is called when the Approve button is clicked on the BP form.

- **Options**

```
Option Public
Option Explicit
Uselsx "*LSXODBC"
Use "eBoatFuncLib"
```

- **Declaration**

```
Const STD_ERROR = 699
Dim agent As NotesAgent
Dim NoteID As String
Dim BPNumQry As ODBCQuery
Dim BPNumRes As ODBCResultSet
Dim CurMax As String
Dim NextMax As String
Dim BPPsw As String
Dim BPProfile As NotesDocument
Dim rtitem As NotesRichTextItem
Dim ProfileDoc As NotesDocument
Dim NewDoc As NotesDocument
Dim st As Integer

Dim session As NotesSession
Public con As ODBCConnection
Public qry As ODBCQuery
Public result As ODBCResultSet
Dim doc As NotesDocument
Dim agentLog As NotesLog
Dim db As NotesDatabase
Dim view As NotesView
Dim dataSource As String
Dim userName As String
Dim password As String
```

- **Initialize**

```
    On Error Goto errorHandler

    Set session = New NotesSession
    Set Agent = session.CurrentAgent
    Set db = session.CurrentDatabase
    NoteID = Agent.ParameterDocID

    Set doc = db.GetDocumentByID(NoteID)

    Set con = New ODBCConnection
    con.Autocommit = False

    dataSource="AS20"
    userName="USERID"
    password="PASSWORD"

    st=con.Connectto(dataSource, userName, password)

    'Create the BP number for the new BP
    Set BPNumQry = New ODBCQuery
    Set BPNumRes = New ODBCResultSet
    Set BPNumQry.Connection = con
    Set BPNumRes.Query = BPNumQry
    BPNumQry.SQL = "SELECT Max(BPNUMBER) as MAXBPNUM FROM eboats.bpinfo"
    BPNumRes.Execute
```

```
    While Not ( BPNumRes.IsEndOfData)
        BPNumRes.NextRow
        CurMax = BPNumRes.GetValue("MAXBPNUM", CurMax)
    Wend
    BPNumRes.Close(DB_CLOSE)
    NextMax = GetNextNumber(CurMax)

    ' Create a new BP record in the database
    If(Not(doc Is Nothing)) Then

        Set qry = New ODBCQuery
        Set result = New ODBCResultSet
        Set qry.Connection = con
        Set result.Query = qry

        qry.SQL = "SELECT * FROM eboats.bpinfo"
        result.Execute

        result.AddRow
        Call result.SetValue("COMPANY", doc.COMPANY(0))
        Call result.SetValue("ADDRESS1",doc.ADDRESS1(0))
        Call result.SetValue("ADDRESS2",doc.ADDRESS2(0))
        Call result.SetValue("CITY", doc.City(0))
        Call result.SetValue("STATE", doc.STATE(0))
        Call result.SetValue("ZIP", doc.ZIP(0))
        Call result.SetValue("COUNTRY", doc.COUNTRY(0))
        Call result.SetValue("PHONE", doc.PHONE(0))
        Call result.SetValue("CFNAME", doc.CFName(0))
        Call result.SetValue("CLNAME", doc.CLName(0))
        Call result.SetValue("EMAIL", doc.EMAIL(0))
        Call result.SetValue("BPNUMBER", NextMax)
        Call result.SetValue("LOCATION", doc.Location(0))
        result.UpdateRow
        Dim res As Integer
        res = con.Committransactions()
        result.Close(DB_CLOSE)

        'Generate a password for the new BP
        BPPsw = GenPassword(8)

        'Register the user in the NAB
        Call AddToNAB

        'Create a new memo
        Set newDoc = New NotesDocument(db)
        newDoc.Subject = "Approval of Business Partnership:"

        Set rtitem = New NotesRichTextItem( newDoc, "Body")
        Call rtitem.AppendText("Your application to become our Business Partner has been
approved!")
        Call rtitem.AddNewLine(1)
        Call rtitem.AppendText("Please remember your id. and password for the next login.")
        Call rtitem.AddNewLine(1)
        Call rtitem.AppendText("Your id. is: " & NextMax & " and your password is: " &
BPPsw)
        Call rtitem.AddNewLine(1)

        ' Send back to the BP
        newDoc.SendTo = doc.EMail(0)

        newDoc.Form = "Memo"
        Call newDoc.Send(True)

        'end of send
        Call Doc.remove(True)

    End If
    con.Disconnect

    Exit Sub

errorHandler:
    If Not ( con Is Nothing) Then
        con.Disconnect
    End If
    Print Err() & "Error : " & Error$
    Resume
```

- **Subroutine AddToNAB**

```
Dim PersonDb As NotesDatabase
Dim PersonDoc As NotesDocument
Dim GroupDb As NotesDatabase
Dim GroupView As NotesView
Dim GroupDoc As NotesDocument
Dim GroupMembers As NotesItem
Dim GroupName As String

Set ProfileDoc = db.getProfileDocument( "RegistrationProfile")
Set PersonDb = New NotesDatabase ( "", ProfileDoc.RegNameDb(0) )
Set GroupDb = New NotesDatabase ( "" , ProfileDoc.RegGroupDb(0) )

GroupName = ProfileDoc.RegGroup(0)

 ' -- Locate the group document that this person will be added to --
Set GroupView = GroupDb.GetView ( "Groups" )
If GroupView Is Nothing Then Error STD_ERROR, "Unable to find view: Groups"
Set GroupDoc = GroupView.GetDocumentByKey ( GroupName )
If GroupDoc Is Nothing Then Error STD_ERROR, "Unable to find group: " + GroupName

  ' -- Create the new person document --
Set PersonDoc = New NotesDocument ( PersonDb )
PersonDoc.Form = "Person"
PersonDoc.Type = "Person"
PersonDoc.LastName = NextMax
PersonDoc.FullName = NextMax
PersonDoc.MailAddress = Doc.Email(0)
PersonDoc.HTTPPassword = BPPsw

PersonDoc.EmployeeID = Doc.CFName(0) + " " + Doc.CLName(0)
PersonDoc.OfficeStreetAddress = Doc.Address1(0)
PersonDoc.CompanyName = Doc.Company(0)
PersonDoc.OfficeCity = Doc.City(0)
PersonDoc.OfficeState =Doc.State(0)
PersonDoc.OfficeZip = Doc.Zip(0)
PersonDoc.OfficeCountry = Doc.Country(0)
PersonDoc.OfficePhoneNumber = Doc.Phone(0)

Call PersonDoc.ComputeWithForm ( False, False )
Call PersonDoc.Save ( False, True )

  ' -- Add the new person to the appropriate group --
Set GroupMembers = GroupDoc.GetFirstItem ( "Members" )
Call GroupMembers.AppendToTextList ( NextMax )
Call GroupDoc.Save ( False, True )
```

- **Subroutine GetNextNumber**

```
Function GetNextNumber( Byval Num As String ) As String
   Dim temp As Integer
   temp = Val(num) + 1
   GetNextNumber = "" & temp
End Function
```

- **Subroutine GenPassword**

```
Function GenPassword ( Byval Length As Integer) As String
   GenPassword = "1234567"
End Function
```

A.3 LS:DO CheckInventory agent

This is the code behind the Check Using LS:DO button on the InventoryQty form in the eboatsB2B.nsf database. This code does a lookup to the PRODUCTS DB2 UDB for AS/400 table and returns the current inventory quantity for the selected model.

- **Options**

```
Option Public
Option Explicit
Uselsx "*LSXODBC"
```

- **Declarations**

```
Dim session As NotesSession
Dim con As ODBCConnection
Dim qry As ODBCQuery
Dim result As ODBCResultSet
Dim doc As NotesDocument, ModelDoc As NotesDocument
Dim db As NotesDatabase
Dim view As NotesView

Dim dataSource As String
Dim userName As String
Dim password As String
Dim qryString As String
```

- **Initialize**

```
    Set session = New NotesSession
    Set db = session.GetDatabase( "", "eboatsInternet.nsf" )

    Set doc = session.Documentcontext
    If (doc Is Nothing) Then
        Print "cannot get document context"
    End If

    Set view = db.GetView( "(AllProducts)" )

    Set Modeldoc = view.GetDocumentByKey( doc.Model(0) , True )

    Call connectToDB2()
    qryString = "SELECT PRODAVAILABLE FROM EBOATS.PRODUCTS WHERE PRODNUMBER='" +
ModelDoc.ProductNumber(0) + "'"

    qry.SQL = qryString

    If Not result.Execute Then
        Print result.GetExtendedErrorMessage,, result.GetErrorMessage
    Else
        result.NextRow
        doc.Results = result.GetValue( "PRODAVAILABLE" )
    End If

    result.Close(DB_CLOSE)
    con.Disconnect
```

- **End SubConnectToDB2 Subroutine**

```
    Set con = New ODBCConnection
    con.Autocommit = False
    Set qry = New ODBCQuery
    Set result = New ODBCResultSet
    Set qry.Connection = con
    Set result.Query = qry

    dataSource="AS20"
    userName="USERID"
    password="PASSWORD"

    ConResult=con.Connectto(dataSource, userName, password)
```

The agent has the settings shown in Figure 261.

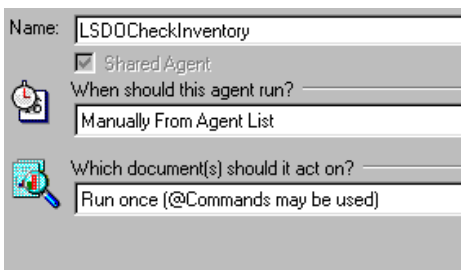


Figure 261. LSDOCheckInventory agent settings

A.4 SelectedBP agent

This agent is contained in the eboatsIntranet.nsf database. This agent is called when a user selects Selected BP from the menu. This agent, which uses LS:DO, queries the ORDERINFO DB2 UDB for AS/400 table, for all orders for the specified business partner, and then prints the results to the display using XML formatting.

- **Options**

```
Option Public
Uselsx "*LSXODBC"
```

- **Declarations**

```
Dim session As NotesSession
Public con As ODBCConnection
Public qry As ODBCQuery
Public result As ODBCResultSet
Dim doc As NotesDocument

Dim agentLog As NotesLog
Dim db As NotesDatabase
Dim view As NotesView

Dim dataSource As String
Dim userName As String
Dim password As String
Dim qryString As String
```

- **Initialize**

```
    Set session = New NotesSession
    Set db = session.CurrentDatabase
    Set doc = db.GetProfileDocument( "ProfileDoc" )

    Call connectToDB2()
    qryString = "SELECT * FROM eboats.ORDERINFO WHERE BPNUMBER=" + "'" & doc.BPNumber(0) &
"'" + ""
    qry.SQL = qryString

    If Not result.Execute Then
        Print result.GetExtendedErrorMessage
        Print result.GetErrorMessage
        Exit Sub
    End If

    'prevents Domino from sending default headers
    Print "Content-type: text/xml"
    Print "<?xml version='1.0' encoding='UTF-8'?>"
    Print "<?xml:stylesheet type='text/xsl' href='ordercatalog.xsl' ?>"
    Print "<ordercatalog>"

    'loop as long as there are elements in the ResultSet
    Do
        result.NextRow

        'send the parent element for each order document
        Print  "<order>"
        Print "<ordernumber>"+result.GetValue("ORDERNUMBER",ordernumber)+"</ordernumber>"
        Print "<orderdate>"+result.GetValue("ORDERDATE",orderdate)+"</orderdate>"
        Print "<bpnumber>"+result.GetValue("BPNUMBER",bpnumber)+"</bpnumber>"
        Print "<shipdate>"+result.GetValue("SHIPDATE",shipdate)+"</shipdate>"
        Print "<prodnumber>"+result.GetValue("PRODNUMBER",prodnumber)+"</prodnumber>"
        Print "<prodcount>"+result.GetValue("PRODCOUNT",prodcount)+"</prodcount>"
        Print "<shipstatus>"+result.GetValue("SHIPSTATUS",shipstatus)+"</shipstatus>"
        Print "<price>"+result.GetValue("PRICE",price)+"</price>"
        'close the order element tag
        Print "</order>"

    Loop Until result.IsEndOfData

     'close the root element tag
    Print "</ordercatalog>"
```

```
result.Close(DB_CLOSE)
con.Disconnect
```

- **Subroutine ConnectToDB2**

```
Set con = New ODBCConnection
con.Autocommit = False
Set qry = New ODBCQuery
Set result = New ODBCResultSet
Set qry.Connection = con
Set result.Query = qry

dataSource="AS20"
userName="USERID"
password="PASSWORD"

ConResult=con.Connectto(dataSource, userName, password)
```

A.5 RegisterCustomer agent

This agent, which uses LS:DO, is contained in the eboatsCustomer.nsf database. This agent is called when a user submits an information request form online.

- **Options**

```
Option Public
Uselsx "*LSXODBC"
Use  "eBoatFuncLib"
```

- **Declarations**

```
Dim nsession As NotesSession
```

- **Initialize**

```
Set nsession = New NotesSession
Set Doc = nsession.DocumentContext

Call connectToDB2()
Call upDateDB2(doc)
Call DisConnectFromDB2()
```

The last subroutines in the code are maintained in a script library called eBoatFuncLib, that can be found in the shared resources in the eboatsCustomer.nsf database. The content of the Script library is shown in Figure 262.

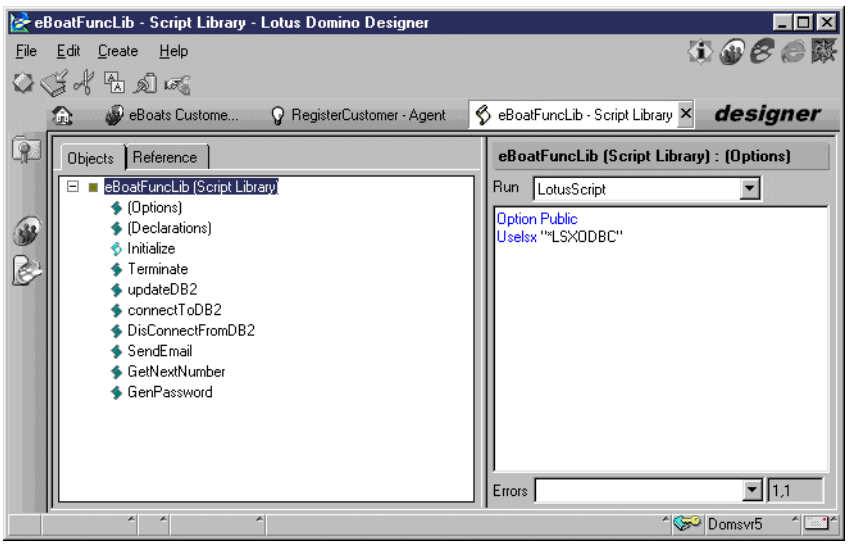


Figure 262. eBoatFuncLib script library in eboatsCustomer.nsf

- **Subroutine ConnectToDB2**

```
    Set con = New ODBCConnection
    con.Autocommit = False
    Set qry = New ODBCQuery
    Set result = New ODBCResultSet
    Set qry.Connection = con
    Set result.Query = qry

    dataSource="AS20"
    userName="USERID"
    password="PASSWORD"

    ConResult=con.Connectto(dataSource, userName, password)
```

- **Subroutine ConnectToDB2**

```
Sub updateDB2(doc As NotesDocument)
    On Error Goto errorHandler

    FormName = doc.Form(0)

    If FormName = "Customer" Then
        qry.SQL = "SELECT * FROM eboats.CUSTOMER"
    Else
        qry.SQL = "SELECT * FROM eboats.BPINFO"
    End If

    result.Execute
    result.AddRow

    Call result.SetValue("ADDRESS1",doc.ADDRESS1(0))
    Call result.SetValue("ADDRESS2",doc.ADDRESS2(0))
    Call result.SetValue("CITY", doc.City(0))
    Call result.SetValue("STATE", doc.STATE(0))
    Call result.SetValue("ZIP", doc.ZIP(0))
    Call result.SetValue("COUNTRY", doc.COUNTRY(0))
    Call result.SetValue("EMAIL", doc.EMAIL(0))
    Call result.SetValue("PHONE", doc.PHONE(0))

    If FormName = "Customer" Then
        Call result.SetValue("FName", doc.FName(0))
        Call result.SetValue("LNAME", doc.LName(0))
    Else
        Call result.SetValue("COMPANY", doc.COMPANY(0))
        Call result.SetValue("CFNAME", doc.CFName(0))
        Call result.SetValue("CLNAME", doc.CLName(0))
    End If
    result.UpdateRow
    Goto TheEnd

errorHandler:

    res = con.Rollbacktransactions()
    'Call agentLog.LogError( Err() , "Error : " & Error$)
    'Call agentLog.Close
    Exit Sub

TheEnd:
    res = con.Committransactions()
    result.Close(DB_CLOSE)

End Sub
```

- **Subroutine DisConnectFromDB2**

```
Sub DisConnectFromDB2
    con.Disconnect
End Sub
```

A.6 LSX LC agent

This is the code behind the Check Using LSX LC button on the InventoryQty form in the eboatsB2B.nsf database. This code does a lookup to the PRODUCTS DB2 UDB for AS/400 table and returns the current inventory quantity for the selected model.

- **Options**

```
Option Public
Option Explicit
Uselsx "*lsxlc"
```

- **Declarations**

```
Dim session As NotesSession
Dim lcsession As LCSession
Dim con As LCConnection
Dim fldLst As LCFieldList
Dim fld As LCField

Dim doc As NotesDocument, ModelDoc As NotesDocument
Dim db As NotesDatabase
Dim view As NotesView

Dim qryString As String
```

- **Initialize**

```
    On Error Goto errorhdl
    Dim agentLog As New NotesLog("agent log")

    Call agentLog.OpenAgentLog()

    Set session = New NotesSession
    Set db = session.GetDatabase( "", "eboatsInternet.nsf" )

    Set doc = session.Documentcontext
    If (doc Is Nothing) Then
        Call agentLog.Logaction( "cannot get document context")
    End If

    Set view = db.GetView( "(AllProducts)" )

    Set Modeldoc = view.GetDocumentByKey( doc.Model(0) , True )

    Set lcsession = New LCSession
    lcsession.ClearStatus

    Set con =  New LCConnection ("db2")
    Set fldLst =  New LCFieldList

  ' set the appropriate properties to connect
    con.Database = "AS20"
    con.Userid = "USERID"
    con.Password = "PASSWORD"
    con.Connect

    qryString = "SELECT PRODAVAILABLE FROM EBOATS.PRODUCTS WHERE PRODNUMBER='" +
ModelDoc.ProductNumber(0) + "'"

    If (con. Execute (qryString,  fldLst) = 0) Then
        Call agentLog.Logaction( "No records were fetched.")
        End
    End If
    Set fld = fldLst.GetField(1)

  ' fetch each record from the result set
    If (con.Fetch (fldLst) > 0) Then
        doc.Results = fld.Text(0)
    End If

    Delete fldlst
    con.Disconnect
    Delete con
```

```
    Call agentLog.close()
    Exit Sub
errorhdl:
    If (lcsession.Status <> LCSUCCESS) Then
        Print lcsession.GetStatusText
    Else
        Print Error$
    End If
```

The agent has the settings shown in Figure 263.

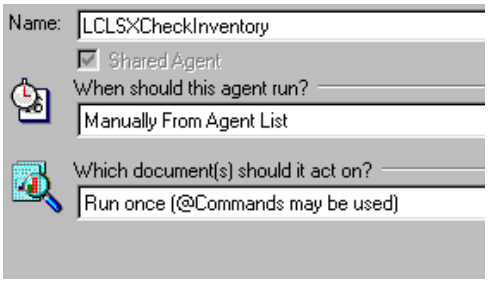


Figure 263. LCLSXCheckInventory agent settings

Appendix B. JavaScript sample code

This appendix lists all the JavaScript code examples presented in the sample databases. Each section is labeled with the name of the function or agent in which the code is contained.

B.1 AboutUs page JSHeader code

This code is contained in the JSHeader for the AboutUs page in the eboatsInternet.nsf database. This code displays a scrolling banner across the bottom of the page when the Web site is accessed:

```
var strText='This database is a sample database developed for IBM Lotus Redbook
\"Developing an e-business Application using Domino for AS/400\".';
var intText=strText.length;
var intSpeed=50;
var intWidth=100;
var intPos=1-intWidth;
function scroll() {
               intPos++;
              var strScroll='';
              if (intPos==intText){
                intPos=1-intWidth;
              }
              if (intPos<0){
                for (var i=1; i<=Math.abs(intPos); i++){
                  strScroll=strScroll+' ';
                }
                strScroll=strScroll+strText.substring(0, intWidth-i+1);
              }
              else{
                strScroll=strScroll+strText.substring(intPos,intWidth+intPos);
              }
              window.status = strScroll;
              setTimeout('scroll()', intSpeed);
            }
function off()
{
           alert('yo')
             window.status = '';
             return true;
}
```

B.2 ShoppingBasket form passthru HTML code

This form is contained in the eboatsB2B.nsf database and is used by Business Partners to place orders. This code is contained directly on the form and is formatted with the text property of passthru HTML:

```
</form><form method=Post  ACTION="/servlet/OrderUpdateServlet">
<table border="1">
<tr>
  <td><b>Model</b></td>
  <td><b>Product Number</b></td>
  <td><b>Qty.</b></td>
  <td><b>Price</b></td>
  <td><b>Total</b></td>
</tr>
<script language="JavaScript">
<!--
grand_total = 0; numItems = 0;
cookies = document.cookie.split(';');
for (i = 0; i < cookies.length; i++) {
      cookie = unescape(cookies[i]).split('=');
      if (cookie.length == 2) {
         params = cookie[1].split('~');
         if (params.length == 4) {
            model = params[0];
            prodNum = params[1];
```

```
            quantity = params[2];
            price = params[3];
            total = quantity * price;
            grand_total += total;
            html = "<tr>"
            html += ("<td>" + model + "</td>");
            html += ("<td>" + prodNum + "</td>");
            html += ("<td>" + quantity + "</td>");
            html += ("<td align=right>" + format(price, 2) + "</td>");
            html += ("<td align=right>" + format(total, 2) + "</td>");
            html += "</tr>";
           html += ("<input type=\"hidden\" name=\"LineItem" + i + "\" value=\"" + cookie[1]
+ "\">");
            document.writeln(html);
            numItems++;
         }
      }
}
if (numItems < 1) {
    document.writeln('<tr><td colspan=5>SHOPPING CART IS EMPTY</td></tr>');
} else {
     document.writeln("<tr><td colspan=5 align=\"right\"><b>$" + format(grand_total, 2) +
"</b></td></tr>");
}
// Create another hidden field indicating the number of line items
document.writeln("<input type=\"hidden\" name=\"LineItemLength\" value=\"" + numItems +
"\">");
// -->
</script>
</table>
<input type="button" value="Clear My Shopping Cart" onClick="empty(this.form);">
<input type=submit value="Place Order">
</form>
```

B.3 ShoppingBasket form JSHeader

This code is contained in the JSHeader for the ShoppingBasket form in the eboatsB2B.nsf database:

```
function setCookie(name, value) {
    document.cookie = name + "=" + escape(value) + "; path=/;";
}

function addProduct() {
     f = document.forms[0];
     ind = f.ProductList.selectedIndex;
    choice =  f.ProductList.options[ind].text ;
    b1 = choice.indexOf( ' ' );
    b2 = choice.lastIndexOf( ' ' ) ;
    model = choice.substring(0,b1);

    prodNum = choice.substring(b1+1,b2);
    quantity = f.Quantity.value;
    if (isNaN(parseInt(quantity)) || parseInt(quantity) < 1) {
        alert('Invalid quantity');
        return;
    }
    price = choice.substring(b2+1, choice.length);
    value = model + "~" + prodNum + "~" + quantity + "~" + price;
    setCookie(model, value);

    window.location = "ShoppingBasket?OpenForm"
}
function empty(form) {
    cookies = document.cookie.split(';');
    for (i = 0; i < cookies.length; i++) {
        cookie = unescape(cookies[i]).split('=');

        if ( cookie[0] != "BPNumber" ) {
                if (cookie.length == 2) {
                cookiename = cookie[0];
                 setCookie(cookiename, '');
             }
         }
    }
```

```
    history.go(0);
}

function format( expr, decplaces ) {
    var str = "" + Math.round( eval( expr ) * Math.pow( 10, decplaces ) );
    while ( str.length <= decplaces ) {
        str = "0" + str;
    }
    var decpoint = str.length - decplaces;
    return str.substring( 0, decpoint ) + "." + str.substring( decpoint, str.length );
}

function dollarize( expr ) {
    return "$" + format( expr, 2 );
}
```

B.4 BP form validate function

This function is contained in the JSHeader of the BP form in the eboatsBP.nsf database. This function validates that the user has entered a value in all required fields and that the information contained in certain fields is not longer than what is supported in the DB2 UDB for AS/400 table:

```
function validate()
{
   var f = document.forms[0];

      if ( f.Company.value == "" ) {
     alert( "You must enter your Company Name" );
        f.Company.focus()
     return false
   }
      if ( f.Address1.value == "" ) {
     alert( "You must enter your Address" );
        f.Address1.focus()
     return false
   }
      if ( f.City.value == "" ) {
     alert( "You must enter your City" );
        f.City.focus()
     return false
   }
      if ( f.State.value == "" ) {
     alert( "You must enter your State or Province" );
        f.State.focus()
     return false
   }
      if ( f.Zip.value == "" ) {
     alert( "You must enter your Postal Code" );
        f.Zip.focus()
     return false
   }
      if ( f.Country.value == "" ) {
     alert( "You must enter your Country" );
        f.Country.focus()
     return false
   }
   if ( f.CFName.value == "" ) {
     alert( "You must enter your First Name" );
     f.CFName.focus()
     return false
   }
   if ( f.CLName.value == "" ) {
     alert( "You must enter your Last Name" );
        f.CLName.focus()
     return false
   }
      if ( f.Email.value == "" ) {
     alert( "You must enter your Email Address" );
        f.Email.focus()
     return false
   }

   if ( f.Company.value.length > 40 ) {
```

```
      alert("Company Name cannot be more than 40 characters.  Please change the Company
Name.")
      f.Company.focus();
      return false;
   }
   if ( f.CFName.value.length > 10 ) {
      alert("The First Name cannot be more than 10 characters.  Please change the First
Name.")
      f.CFName.focus();
      return false;
   }
   if ( f.CLName.value.length > 10 ) {
      alert("The Last Name cannot be more than 10 characters.  Please change the Last Name.")
      f.CLName.focus();
      return false;
   }
   if ( f.Address1.value.length > 20 ) {
      alert("Address1 cannot be more than 20 characters.  Please change Address1.")
      f.Address1.focus();
      return false;
   }
   if ( f.Address2.value.length > 20 ) {
      alert("Address2 cannot be more than 20 characters.  Please change Address2.")
      f.Address2.focus();
      return false;
   }
   if ( f.City.value.length > 20 ) {
      alert("City cannot be more than 20 characters.  Please change City.")
      f.City.focus();
      return false;
   }
   if ( f.State.value.length > 20 ) {
      alert("State or Province cannot be more than 20 characters.  Please change State or
Province.")
      f.State.focus();
      return false;
   }
   if ( f.Zip.value.length > 7 ) {
      alert("The Postal Code cannot be more than 7 characters.  Please change the Postal
Code.")
      f.Zip.focus();
      return false;
   }
   if ( f.Country.value.length > 20 ) {
      alert("Country cannot be more than 20 characters.  Please change Country.")
      f.Country.focus();
      return false;
   }
   if ( f.Email.value.length > 30 ) {
      alert("Email address cannot be more than 30 characters.  Please change Email address.")
      f.Email.focus();
      return false;
   }
    if ( f.Phone.value.length > 20 ) {
      alert("Phone number cannot be more than 20 characters.  Please change Phone number.")
      f.Phone.focus();
      return false;
   }

  window.status = ''
}
```

B.5 Customer Info Request form validate function

This function is contained in the JSHeader of the Customer form in the eboatsCustomer.nsf database. This function validates that the user has entered a value in all required fields and that the information contained in certain fields is not longer than what is supported in the DB2 UDB for AS/400 table.

```
function validate()
{
   var f = document.forms[0];

   if ( f.FName.value == "" ) {
```

```
    alert( "You must enter your First Name" );
    f.FName.focus()
    return false
  }
  if ( f.LName.value == "" ) {
    alert( "You must enter your Last Name" );
        f.LName.focus()
    return false
  }
  if ( f.Phone.value == "" ) {
    alert( "You must enter your Phone Number" );
        f.Phone.focus()
    return false
  }
  if ( f.Email.value == "" ) {
    alert( "You must enter your Email Address" );
        f.Email.focus()
    return false
  }

  if ( f.FName.value.length > 10 ) {
     alert("The First Name cannot be more than 10 characters.  Please change the First
Name.")
     f.FName.focus();
     return false;
  }
  if ( f.LName.value.length > 10 ) {
     alert("The Last Name cannot be more than 10 characters.  Please change the Last Name.")
     f.LName.focus();
     return false;
  }
  if ( f.Address1.value.engths > 20 ) {
     alert("Address1 cannot be more than 20 characters.  Please change Address1.")
     f.Address1.focus();
     return false;
  }
  if ( f.Address2.value.length > 20 ) {
     alert("Address2 cannot be more than 20 characters.  Please change Address2.")
     f.Address2.focus();
     return false;
  }
  if ( f.City.value.length > 20 ) {
     alert("City cannot be more than 20 characters.  Please change City.")
     f.City.focus();
     return false;
  }
  if ( f.State.value.length > 20 ) {
     alert("State or Province cannot be more than 20 characters.  Please change State or
Province.")
     f.State.focus();
     return false;
  }
  if ( f.Zip.value.length > 7 ) {
     alert("The Postal Code cannot be more than 7 characters.  Please change the Postal
Code.")
     f.Zip.focus();
     return false;
  }
  if ( f.Country.value.length > 20 ) {
     alert("Country cannot be more than 20 characters.  Please change Country.")
     f.Country.focus();
     return false;
  }
  if ( f.Email.value.length > 30 ) {
     alert("Email address cannot be more than 30 characters.  Please change Email address.")
     f.Email.focus();
     return false;
  }
   if ( f.Phone.value.length > 20 ) {
     alert("Phone number cannot be more than 20 characters.  Please change Phone number.")
     f.Phone.focus();
     return false;
  }

 window.status = ''
 f.submit();
}
```

Appendix C. Java sample code

This appendix lists all the Java code examples presented in the sample databases. Each section is labeled with the name of the function or agent in which the code is contained.

The code examples are divided in servlets and JavaBeans.

C.1 Java servlets

This section describes all of the Java servlets used in the sample application.

C.1.1 OrderQueryServlet servlet

This servlet is stored on the AS/400 system in the domino\servlet directory, found in the Domino data directory. This servlet is called when a business partner clicks Order Status from the menu in the eboatsB2B.nsf database. This servlet reads a cookie to get the business partner number, and passes this information to the OrderInfoQuery JavaBean, detailed in C.2.1, “OrderInfoQuery JavaBean” on page 350.

```
package eboats;

import java.io.*;
import java.rmi.*;
import java.util.*;
import java.lang.*;
import java.math.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Insert the type's description here.
 * Creation date: (2/21/00 10:59:46 AM)
 * @author: Eric
 */
public class OrderQueryServlet extends javax.servlet.http.HttpServlet  {
    private OrderInfoQuery orderQueryBean;
    private final String USR = USERID;
    private final String PWD = PASSWORD;
/**
 * OrderQueryServlet constructor comment.
 */
public OrderQueryServlet() {
    super();
}
/**
 * This method was created in VisualAge.
 */
public void destroy() {

}
/**
 * This method was created in VisualAge.
 * @param out java.io.PrintWriter
 * @param qb eboats.OrderInfoQuery
 */
public void displayResults(PrintWriter out, OrderInfoQuery qbean)
throws java.sql.SQLException{

    out.println("<table border>");
    out.println("<caption><h2>Order Status</h2></caption>");
    // print out header
    out.print("<tr><h3>");
    out.print("<th>Order Number</th>");
    out.print("<th>Order Date</th>");
    out.print("<th>Product Number</th>");
    out.print("<th>Order Quantity</th>");
```

```
    out.print("<th>Unit Price</th>");
    out.print("<th>Shipping Date</th>");
    out.print("<th>Shipping Status</th>");
    out.print("</h3></tr>");

    //print out content
    while (qbean.next()){
        out.print("<tr>");
        out.print("<td>" + qbean.getOrderNumber()+"</td>");
        out.print("<td>" + qbean.getOrderDate()+"</td>");
        out.print("<td>" + qbean.getProdNumber() + "</td>");
        out.print("<td>" + qbean.getProdCount() + "</td>");
        out.print("<td>" + qbean.getPrice() + "</td>");
        out.print("<td>" + qbean.getShipDate() + "</td>");
        out.print("<td>" + qbean.getShipStatus() + "</td>");
        out.print("</tr>");
    }
    out.println("</table>");

}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 11:02:13 AM)
 * @param request com.sun.server.http.HttpServiceRequest
 * @param response com.sun.server.http.HttpServiceResponse
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws java.io.IOException
{

    // Set the content type to HTML
    response.setContentType("text/html");

    // Prevent caching of dynamic data
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");

    // Assemble top and bottom of HTML page
    String top = "<html><head><title>OrderQueryServlet</title></head><body><center>";
    String bottom = "</center></body></html>";

    // Print the top of the HTML page
    PrintWriter out = response.getWriter();
    out.println(top);

//  String orderNumber = request.getParameterValues("OrderNumber")[0];

    Cookie[] cookies = request.getCookies();
    String bpNumber = "";

    if ( cookies == null)
        out.println("cookies is null");
    else {
        for ( int i=0; i< cookies.length; i++) {
            if (cookies[i].getName().equals(new String("BPNumber"))){
                bpNumber = cookies[i].getValue();
                out.println("<h1> Your BP number is:<i>" + bpNumber + "</i></h1>");
            }
        }
    }

    try {
      // instantiate the beans and store them so they can be accessed by the called page

        //Connect to the database
        orderQueryBean = new OrderInfoQuery();
        orderQueryBean.connectToDB(USR, PWD);
        orderQueryBean.excecute(bpNumber);

        //display the results
        displayResults(out, orderQueryBean);

        orderQueryBean.closeConnect();
        out.println("<p><h2>Thank you for your order.</h2><br>");

    } catch (Exception e) {
        out.println("<h1>Error</h1>");
```

```
       out.println("<h2>An error occurred. Please try again, or forward this page to
webmaster@acme.com.</h2>");
       out.println(e.getMessage() + "<p>");
       e.printStackTrace(out);
    }

    // Print the bottom of the HTML page
    out.println(bottom);

}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 11:02:13 AM)
 * @param request com.sun.server.http.HttpServiceRequest
 * @param response com.sun.server.http.HttpServiceResponse
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws java.io.IOException
{

    // Set the content type to HTML
    response.setContentType("text/html");

    // Prevent caching of dynamic data
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");

    // Assemble top and bottom of HTML page
    String top = "<html><head><title>OrderQueryServlet</title></head><body><center>";
    String bottom = "</center></body></html>";

    // Print the top of the HTML page
    PrintWriter out = response.getWriter();
    out.println(top);

    out.println("begin post");
//  String orderNumber = request.getParameterValues("OrderNumber")[0];

    Cookie[] cookies = request.getCookies();
    String bpNumber = "";

    if ( cookies == null)
        out.println("cookies is null");
    else {
        for ( int i=0; i< cookies.length; i++) {
            out.println(" cookie:" + cookies[i].getName());
            out.println(" cookie values:" + cookies[i].getValue() + "<br>");
            if (cookies[i].getName() == "BPNumber")
                bpNumber = cookies[i].getValue();
        }
    }

    try {
      // instantiate the beans and store them so they can be accessed by the called page

        //Connect to the database
        orderQueryBean = new OrderInfoQuery();
        orderQueryBean.connectToDB(USR, PWD);
        orderQueryBean.excecute(bpNumber);

        //display the results
        displayResults(out, orderQueryBean);

        orderQueryBean.closeConnect();
        out.println("the order has been loaded. Thanks. <br>");

    } catch (Exception e) {
        out.println("<h1>Error</h1>");
        out.println("<h2>An error occurred. Please try again, or forward this page to
webmaster@acme.com.</h2>");
        out.println(e.getMessage() + "<p>");
        e.printStackTrace(out);
    }

    // Print the bottom of the HTML page
    out.println(bottom);
}
/**
```

```
 * Insert the method's description here.
 * Creation date: (2/21/00 1:06:24 PM)
 * @param config javax.servlet.ServletConfig
 */
public void init(ServletConfig config)
throws javax.servlet.ServletException
{
    super.init(config);
}
/**
 * This method was created in VisualAge.
 * @return java.lang.String
 */
public String today() {
    Calendar cal = Calendar.getInstance();
    String temp =
String.valueOf(cal.get(Calendar.YEAR))+String.valueOf(cal.get(Calendar.MONTH)) +
String.valueOf(cal.get(Calendar.DATE));
    return temp;
}
}
```

C.1.2 OrderUpdateServlet Java servlet

This servlet is stored on the AS/400 system in the domino\servlet directory found in the Domino data directory. This servlet is called when a business partner submits orders in the eboatsB2B database. This servlet reads a cookie to get the business partner number and passes this information with the details of the order to the OrderInfoInput JavaBean, detailed in C.2.2, “OrderInfoInput JavaBean” on page 353.

```
package eboats;

import java.io.*;
import java.rmi.*;
import java.util.*;
import java.lang.*;
import java.math.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Insert the type's description here.
 * Creation date: (2/21/00 10:59:46 AM)
 * @author: Eric
 */
public class OrderUpdateServlet extends javax.servlet.http.HttpServlet  {
    private OrderInfoInput orderInputBean;
    private final String USR = USERID;
    private final String PWD = PASSWORD;
    private PrintWriter out;
/**
 * OrderUpdateServlet constructor comment.
 */
public OrderUpdateServlet() {
    super();
}
/**
 * This method was created in VisualAge.
 */
public void destroy() {

}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 11:02:13 AM)
 * @param request com.sun.server.http.HttpServiceRequest
 * @param response com.sun.server.http.HttpServiceResponse
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws java.io.IOException
{
    HttpSession session = request.getSession(false);

    // Set the content type to HTML
```

```
    response.setContentType("text/html");

    // Prevent caching of dynamic data
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");

    // Assemble top and bottom of HTML page
    String top = "<html><head><title>OrderUpdateServlet</title></head><body><center>";
    String bottom = "</center></body></html>";

    // Print the top of the HTML page
    PrintWriter out = response.getWriter();
    out.println(top);

    out.println("begin Get");

    try {
    out.println("begin1");
      // instantiate the beans and store them so they can be accessed by the called page
    String[] names = session.getValueNames();
    for (int i=0; i< names.length; i++){
        out.println(names[i]+":" + session.getValue(names[i]) + "<BR>");
    }



    } catch (Exception e) {
        out.println("<h1>Error</h1>");
        out.println("<h2>An error occurred. Please try again, or forward this page to
webmaster@acme.com.</h2>");
        out.println(e.getMessage() + "<p>");
        e.printStackTrace(out);
    }

    // Print the bottom of the HTML page
    out.println(bottom);


}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 11:02:13 AM)
 * @param request com.sun.server.http.HttpServiceRequest
 * @param response com.sun.server.http.HttpServiceResponse
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws java.io.IOException
{

    // Set the content type to HTML
    response.setContentType("text/html");

    // Prevent caching of dynamic data
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");

    // Assemble top and bottom of HTML page
    String top = "<html><head><title>OrderUpdateServlet</title></head><body><center>";
    String bottom = "</center></body></html>";

    // Print the top of the HTML page
    out = response.getWriter();
    out.println(top);

    try {
      // instantiate the beans and store them so they can be accessed by the called page

    Cookie[] cookies = request.getCookies();
    if ( cookies == null)
        out.println("cookies is null");
    else {

        //Connect to the database
        orderInputBean = new OrderInfoInput();
        orderInputBean.connectToDB(USR, PWD);

        String bpNumber = "";
        String model;
```

```
        BigDecimal price;
        String productNumber;
        BigDecimal quantity;

        // save the order information to the db2 table
        for ( int i=0; i< cookies.length; i++) {
            if (cookies[i].getName().trim().equals( new String("BPNumber"))){
                bpNumber = cookies[i].getValue();
                out.println("<h1><b>Your BP number is:</b> <I>" +
bpNumber+"</I></h1><br><p></p>");
            }
        }
        out.println("<table border>");
        out.println("<caption><h2>Order List</h2></caption>");
        // print out header
        out.print("<tr><h3>");
        out.print("<th>Order Number</th>");
        out.print("<th>Model</th>");
        out.print("<th>Product Number</th>");
        out.print("<th>Order Quantity</th>");
        out.print("<th>Unit Price</th>");
        out.print("</h3></tr>");

        for ( int i = 0; i < cookies.length; i++){
            if ((!cookies[i].getName().trim().equals(new
String("DomAuthSessId")))&&(!cookies[i].getName().trim().equals(new String("BPNumber"))) &&
(cookies[i].getValue() != null)){
             String tempValue = cookies[i].getValue();
             System.out.println(tempValue);
             model = getModel(tempValue);
             price = getPrice(tempValue);
                productNumber = getProductNumber(tempValue);
                quantity = getQuantity(tempValue);

                orderInputBean.setBpNumber(bpNumber);
                orderInputBean.setOrderDate(today());
                String ordnum = orderInputBean.getOrderNumber();
                orderInputBean.setPrice(price);
                orderInputBean.setProdCount(quantity);
                orderInputBean.setProdNumber(productNumber);
                orderInputBean.setShipDate("");
                orderInputBean.setShipStatus("0");
                orderInputBean.excecute();
                out.println("<tr>");
                 out.println("<td>" +  ordnum + "</td>");
                 out.println("<td>"+ model + "</td>");
                 out.println("<td>" + productNumber + "</td>");
                 out.println("<td>" + quantity.toString() + "</td>");
                 out.println("<td>" + price.toString() + "</td>");
                out.println("</tr>");
            }
        }
        out.print("</table>");
        out.println("</h2><h1><b>Thank you very much!</b></h1>");
        orderInputBean.closeConnect();
    }



    } catch (Exception e) {
        out.println("<h1>Error</h1>");
        out.println("<h2>An error occurred. Please try again, or forward this page to
webmaster@acme.com.</h2>");
        out.println(e.getMessage() + "<p>");
        e.printStackTrace(out);
    }

    // Print the bottom of the HTML page
    out.println(bottom);


}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 6:32:16 PM)
 * @return java.lang.String
 */
 public String getModel(String ivalue) {
```

```
     int end;
     int start;

     String temp;
     start = 0;
     System.out.println("the mode input value is:" + ivalue);
     end = ivalue.indexOf("%7E", 1);
     temp = ivalue.substring(start, end);
     return temp;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 6:34:03 PM)
 * @return java.math.BigDecimal
 */
public java.math.BigDecimal getPrice(String iValue) {
    String temp;
    BigDecimal res;
    int start;
    int end;

    start = iValue.lastIndexOf("%7E")+3;
    end = iValue.length();
    temp = iValue.substring(start, end);
    res = new BigDecimal(temp);
    return res;

}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 6:32:45 PM)
 * @return java.lang.String
 */
public String getProductNumber(String ivalue) {
    String temp;
    int start;
    int end;

    start = ivalue.indexOf("%7E")+3;
    end = ivalue.indexOf("%7E", start+1);
    temp = ivalue.substring(start, end);
    return temp;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 6:33:18 PM)
 * @return java.math.BigDecimal
 */
public java.math.BigDecimal getQuantity(String ivalue) {
    String temp;
    int start;
    int end;

    end = ivalue.lastIndexOf("%7E");
    start = ivalue.lastIndexOf("%7E", end-1)+3;
    temp = ivalue.substring(start, end);
    BigDecimal res= new BigDecimal(temp);
    return res;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 1:06:24 PM)
 * @param config javax.servlet.ServletConfig
 */
public void init(ServletConfig config)
throws javax.servlet.ServletException
{
    super.init(config);
}
/**
 * This method was created in VisualAge.
 * @return java.lang.String
 */
public String today() {
    Calendar cal = Calendar.getInstance();
    int mon = cal.get(Calendar.MONTH)+1;
    int dat = cal.get(Calendar.DATE);
    String month;
```

```
    String date;
    if (mon < 10){
        StringBuffer tmp = new StringBuffer("0");
        tmp.append(String.valueOf(mon));
        month = tmp.toString();
    }else
         month = String.valueOf(mon);

    if (dat < 10){
        StringBuffer tmp = new StringBuffer("0");
        tmp.append(String.valueOf(dat));
        date = tmp.toString();
    }else
         date = String.valueOf(dat);

    String temp = String.valueOf(cal.get(Calendar.YEAR))+ month + date;
    return temp;
```

C.2 JavaBeans

This section lists all JavaBeans contained in the sample application.

C.2.1 OrderInfoQuery JavaBean

This JavaBean is stored on the AS/400 system in the domino\servlet directory found in the Domino data directory. This JavaBean is called by the OrderQueryServlet Java servlet. This JavaBean queries the ORDERINFO DB2 UDB for AS/400 table to get all orders for the specified business partner number. It returns each record for all orders to the OrderQueryServlet Java servlet.

```
package eboats;

import java.util.*;
import java.lang.*;
import java.sql.*;
import java.net.*;
import java.math.*;

/**
 * Insert the type's description here.
 * Creation date: (2/21/00 5:14:29 PM)
 * @author: Eric
 */
public class OrderInfoQuery {
    private String bpNumber;
    private String orderDate;
    private String orderNumber;
    private java.math.BigDecimal prodCount;
    private String prodNumber;
    private String shipDate;
    private String shipStatus;
    private java.math.BigDecimal price;
// If you try to access your system, you should change the SYSTEMNAME to your local database
name in the relational database directory.
    private final String SYSTEMNAME = "as20";
    private final String LIB = "EBOATS";
//  private String statement = "SELECT ORDERNUMBER, ORDERDATE, BPNUMBER, SHIPDATE, PRODNUMBER,
PRODCOUNT, SHIPSTATUS, PRICE FROM EBOATS.ORDERINFO WHERE BPNUMBER = ?";
    private String statement = "call eboats.queryorder(?)";
    private java.sql.Connection dbConnection;
//  private java.sql.PreparedStatement psQueryOrder;
    private CallableStatement psQueryOrder;
    private ResultSet res;
/**
 * OrderInfoQuery constructor comment.
 */
public OrderInfoQuery() {
    super();
}
/**
 * This method was created in VisualAge.
 */
public void closeConnect() {
```

```
    try{
    dbConnection.close();
    } catch ( java.sql.SQLException e){
    }
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:15:37 PM)
 * @param userid java.lang.String
 * @param password java.lang.String
 */
public void connectToDB(String userid, String password) {
    // Create a properties object for JDBC connection
    Properties jdbcProperties = new Properties();

    // Set the properties for the JDBC connection
    jdbcProperties.put("user", userid);
    jdbcProperties.put("password", password);
    jdbcProperties.put("naming", "sql");
    jdbcProperties.put("errors", "full");
    jdbcProperties.put("date format", "iso");
    //jdbcProperties.put("extended dynamic", "true");
    //jdbcProperties.put("package", "SerTest");
    try {
        netscape.security.PrivilegeManager.enablePrivilege("UniversalConnect"); // Needed for
Netscape URL
    } catch (Throwable exception) {
        System.out.println("error in enable security for Netscape");
    }
    try {
        System.out.println("before register driver");
        DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver()); //Toolbox
JDBC driver

        System.out.println("before getConnection");
        dbConnection = DriverManager.getConnection("jdbc:as400://" + SYSTEMNAME + "/" + LIB,
jdbcProperties); //toolbox

    } catch (SQLException ex) {
        System.out.println("connect failed");
        ex.printStackTrace();
        return;
    }
    System.out.println("connected successfully");
    return;

}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:25:38 PM)
 */
public void excecute(String iNum) {
    try {
        System.out.println("before prepare");

//      psQueryOrder = dbConnection.prepareStatement(statement);
        psQueryOrder = dbConnection.prepareCall(statement);
        System.out.println("the input iNum is:");
        System.out.println(iNum);
        psQueryOrder.setString(1, iNum);

        res = psQueryOrder.executeQuery();

    }catch ( java.sql.SQLException e){

    }
}
/**
 * Insert the method's description here.
 * Creation date: (2/22/00 3:46:41 PM)
 * @param idate java.lang.String
 */
private String formatDate(String idate) {
    StringBuffer sbf = new StringBuffer(11);
    sbf.append(idate.substring(0,4));
    sbf.append(".");
    sbf.append(idate.substring(4,6));
    sbf.append(".");
```

```
    sbf.append(idate.substring(6));
    return sbf.toString();
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:19:29 PM)
 * @param inum java.lang.String
 */
public String getBpNumber() {
   return bpNumber;
}
/**
 * This method was created in VisualAge.
 * @return java.lang.String
 * @param i int
 */
public String getFieldName(int i) throws java.sql.SQLException{
    ResultSetMetaData resMeta = res.getMetaData();
    String temp = resMeta.getColumnName(i);
    return temp;
}
/**
 * This method was created in VisualAge.
 * @param fname java.lang.String
 */
public int getFieldSize(String fname) throws java.sql.SQLException {
    int tempsize = 0;
    ResultSetMetaData resMeta = res.getMetaData();
    int cols = resMeta.getColumnCount();
    for (int i=1; i <= cols; i++) {
        if ( fname == resMeta.getColumnName(i).trim()){
            tempsize = resMeta.getColumnDisplaySize(i);
            break;
        }
    }
    return tempsize;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:18:50 PM)
 * @param idate java.lang.String
 */
public String getOrderDate() {
     return orderDate;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:17:57 PM)
 * @param num java.lang.String
 */
public String getOrderNumber() {
   return orderNumber;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:21:34 PM)
 * @param param java.math.BigDecimal
 */
public String getPrice() {
    return price.toString();
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:21:34 PM)
 * @param param java.math.BigDecimal
 */
public String getProdCount() {
    return prodCount.toString();
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:20:49 PM)
 * @param inum java.lang.String
 */
public String getProdNumber() {
    return prodNumber;
}
/**
```

```
 * Insert the method's description here.
 * Creation date: (2/21/00 5:20:18 PM)
 * @param idate java.lang.String
 */
public String getShipDate() {
    return shipDate;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:22:03 PM)
 * @param istatus java.lang.String
 */
public String getShipStatus() {
    return shipStatus;
}
/**
 * This method was created in VisualAge.
 * @return java.lang.Boolean
 */
public boolean next() throws java.sql.SQLException{
    if (res.next()){
        bpNumber = res.getString("BPNUMBER");
        System.out.println("bpnumber is:");
        System.out.println(bpNumber);

        orderDate = formatDate(res.getString("ORDERDATE"));
        orderNumber = res.getString("ORDERNUMBER");
        price = res.getBigDecimal("PRICE",2);
        prodCount = res.getBigDecimal("PRODCOUNT",2);
        prodNumber = res.getString("PRODNUMBER");
        shipDate = formatDate(res.getString("SHIPDATE"));
        shipStatus = res.getString("SHIPSTATUS");
        if (shipStatus.equals(new String("0")))
            shipStatus = "loaded";
        else if (shipStatus.equals(new String("1")))
            shipStatus = "shipping";
        else
            shipStatus = "received";

        return true;
    }else
        return false;
}
}
```

C.2.2 OrderInfoInput JavaBean

This JavaBean is stored on the AS/400 system in the domino\servlet directory found in the Domino data directory. This JavaBean is called by the OrderUpdateServlet Java servlet. This JavaBean creates new order records in the ORDERINFO DB2 UDB for AS/400 table.

```
package eboats;

import java.util.*;
import java.lang.*;
import java.sql.*;
import java.net.*;
import java.math.*;

/**
 * Insert the type's description here.
 * Creation date: (2/21/00 5:14:29 PM)
 * @author: Eric
 */
public class OrderInfoInput {
    private String bpNumber;
    private String orderDate;
    private String orderNumber;
    private java.math.BigDecimal prodCount;
    private String prodNumber;
    private String shipDate;
    private String shipStatus;
    private java.math.BigDecimal price;
// if you run the JavaBean on your system, you should modify the SYSTEMNAME to the local
database name in your relational database directory.
```

```
    private final String SYSTEMNAME = "as20";
    private final String LIB = "EBOATS";
    private String statement = "INSERT INTO EBOATS.ORDERINFO (ORDERNUMBER, ORDERDATE,
BPNUMBER, SHIPDATE, PRODNUMBER, PRODCOUNT, SHIPSTATUS, PRICE) VALUES(?,?,?,?,?,?,?,?)";
    private java.sql.Connection dbConnection;
    private java.sql.PreparedStatement psInputOrder;
/**
 * OrderInfoInput constructor comment.
 */
public OrderInfoInput() {
    super();
}
/**
 * This method was created in VisualAge.
 */
public void closeConnect() {
    try{
    dbConnection.close();
    } catch ( java.sql.SQLException e){
    }
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:15:37 PM)
 * @param userid java.lang.String
 * @param password java.lang.String
 */
public void connectToDB(String userid, String password) {
    // Create a properties object for JDBC connection
    Properties jdbcProperties = new Properties();

    // Set the properties for the JDBC connection
    jdbcProperties.put("user", userid);
    jdbcProperties.put("password", password);
    jdbcProperties.put("naming", "sql");
    jdbcProperties.put("errors", "full");
    jdbcProperties.put("date format", "iso");
    //jdbcProperties.put("extended dynamic", "true");
    //jdbcProperties.put("package", "SerTest");
    try {
        netscape.security.PrivilegeManager.enablePrivilege("UniversalConnect"); // Needed for
Netscape URL
    } catch (Throwable exception) {
        System.out.println("error in enable security for Netscape");
    }
    try {
        System.out.println("before register driver");
        DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver()); //Toolbox
JDBC driver

        System.out.println("before getConnection");
        dbConnection = DriverManager.getConnection("jdbc:as400://" + SYSTEMNAME + "/" + LIB,
jdbcProperties); //toolbox

    } catch (SQLException ex) {
        System.out.println("connect failed");
        ex.printStackTrace();
        return;
    }
    System.out.println("connected successfully");
    return;

}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:25:38 PM)
 */
public void excecute() {
    try {


        //psAllRecord = dbConnection.prepareStatement("select * from apilib.parts");
        psInputOrder = dbConnection.prepareStatement(statement);

        psInputOrder.setString(1, orderNumber);
        psInputOrder.setString(2, orderDate);
        psInputOrder.setString(3, bpNumber);
        psInputOrder.setString(4, shipDate);
```

```
        psInputOrder.setString(5, prodNumber);
        psInputOrder.setBigDecimal(6, prodCount);
        psInputOrder.setString(7, shipStatus);
        psInputOrder.setBigDecimal(8, price);
        System.out.println("the shipDate is:");
        System.out.println(shipDate);
        System.out.println("the orderNumber is:");
        System.out.println(orderNumber);
        psInputOrder.executeUpdate();
        System.out.println("end of execution.");
    }catch ( java.sql.SQLException e){
         System.out.println(e.getMessage());
    }
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:25:38 PM)
 */
public void excecute(java.io.PrintWriter out) {
    try {
        System.out.println("before prepare");

        //psAllRecord = dbConnection.prepareStatement("select * from apilib.parts");
        out.println("the statement is:" + statement + "<br>");
        psInputOrder = dbConnection.prepareStatement(statement);

        out.println("the orderNumber is:" + orderNumber+"<br>");
        psInputOrder.setString(1, orderNumber);

        out.println("the orderDate is:" + orderDate +"<br>");
        psInputOrder.setString(2, orderDate);

        out.println("the bpNumber is:" + bpNumber + "<br>");
        psInputOrder.setString(3, bpNumber);

        out.println("the shipDate is:" + shipDate+"<br>");

        psInputOrder.setString(4, shipDate);
        out.println("the prodNumber is:" + prodNumber+"<br>");
        psInputOrder.setString(5, prodNumber);

        out.println("the prodCount is:" + prodCount.toString() + "<br>");
        psInputOrder.setBigDecimal(6, prodCount);

        out.println("the shipStatus is:" + shipStatus +"<br>");
        psInputOrder.setString(7, shipStatus);
        out.println("the price is:" + price.toString() +"<br>");
        psInputOrder.setBigDecimal(8, price);

        psInputOrder.executeUpdate();

    }catch ( java.sql.SQLException e){

    }
}
/**
 * This method was created in VisualAge.
 * @return int
 * This method query the orderinfo database to get max order number and generate a new one.
 * returning null means not successful, otherwise successfully created a new one.
 */
public String getOrderNumber() {
    try{
        String qryNumStm = "Select max(ordernumber) as maxnum from eboats.orderinfo";
        orderNumber = "";
        PreparedStatement maxQry = dbConnection.prepareStatement(qryNumStm);
        ResultSet res = maxQry.executeQuery();
        ResultSetMetaData resMeta = res.getMetaData();
        int len = resMeta.getColumnDisplaySize(1);
//      System.out.println("the length of the column display size is:" +
Integer.toString(len));
        if (res.next()){
            orderNumber = res.getString(1);
            if ( res.wasNull()){
                StringBuffer tmpBuf = new StringBuffer(len+1);
                for (int i = 0; i< len-1; i++)
                    tmpBuf.append('0');
                tmpBuf.append('1');
```

```
                orderNumber = tmpBuf.toString();
            }else{
//              System.out.println("the orderNumber is:" + orderNumber);
//              System.out.print("the length of ordernumber is:");
//              System.out.println(orderNumber.length());
                orderNumber = orderNumber.trim();
//              System.out.print("now the length of ordernumber is:");
//              System.out.println( orderNumber.length());
                int tmp = Integer.valueOf(orderNumber).intValue() + 1;
                orderNumber = String.valueOf(tmp);
//              System.out.println("the new orderNumber is:" + orderNumber);
                if (orderNumber.length() < len ){
                    StringBuffer buf = new StringBuffer(len+1);
                    for ( int i = 0; i< len - orderNumber.length(); i++)
                        buf.append('0');
                    buf.append(orderNumber);
                    orderNumber = buf.toString();
                }
            }
        }else {
                StringBuffer tmpBuf = new StringBuffer(len+1);
                for (int i = 0; i< len-1; i++)
                    tmpBuf.append('0');
                tmpBuf.append('1');
                orderNumber = tmpBuf.toString();
        }
        res.close();
    }catch (java.sql.SQLException e){
        System.out.println(e.getMessage());
        return null;
    }

    return orderNumber;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:19:29 PM)
 * @param inum java.lang.String
 */
public void setBpNumber(String inum) {
   bpNumber = inum;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:18:50 PM)
 * @param idate java.lang.String
 */
public void setOrderDate(String idate) {
     orderDate = idate;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:17:57 PM)
 * @param num java.lang.String
 */
public void setOrderNumber(String inum) {
   orderNumber = inum;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:21:34 PM)
 * @param param java.math.BigDecimal
 */
public void setPrice(java.math.BigDecimal iprice) {
    price = iprice;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:21:34 PM)
 * @param param java.math.BigDecimal
 */
public void setProdCount(java.math.BigDecimal iprodCount) {
    prodCount = iprodCount;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:20:49 PM)
 * @param inum java.lang.String
```

```
 */
public void setProdNumber(String inum) {
    prodNumber = inum;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:20:18 PM)
 * @param idate java.lang.String
 */
public void setShipDate(String idate) {
    shipDate = idate;
}
/**
 * Insert the method's description here.
 * Creation date: (2/21/00 5:22:03 PM)
 * @param istatus java.lang.String
 */
public void setShipStatus(String istatus) {
    shipStatus = istatus;
}
}
```

C.3 Deploying the servlet on Domino R5.0

Here we describe how to setup the servlet environment on Domino R5.0. We also give you the details about how to deploy the above servlets to Domino R5.0.

Servlets are loaded and called by the Domino Java Servlet Manager, a part of the HTTP server task. The runtime Java support for servlets is provided by the Domino Java Virtual Machine (JVM). When the HTTP task is started, it can automatically start the servlet manager and load the JVM. The HTTP task writes status messages for these operations to the server console and log file.

The servlet manager is controlled by settings in the Domino Directory server document. The settings are located on the **Internet Protocols->Domino Web Engine** tab of the **Server** document. Figure 264 shows the configuration.

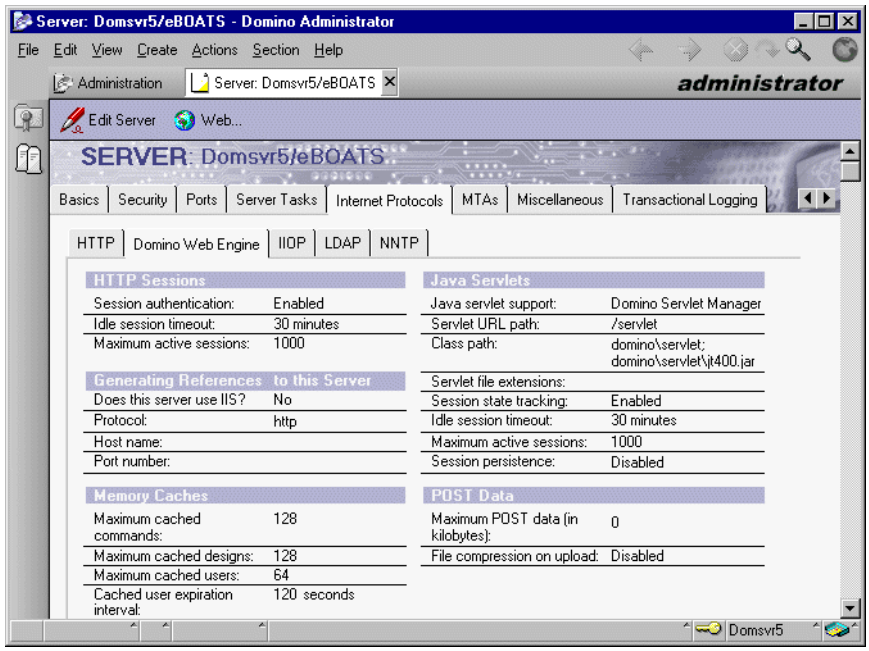


Figure 264. Domino Servlet Manager configuration

A description of the configuration settings is provided in Table 35.

Table 35. Java servlet support configuration parameters

Setting	Options
Java servlet support	**None**: the default value. The HTTP task does not load the servlet manager or the JVM. **Domino Servlet Manager**: The HTTP task loads both the JVM and the servlet manager. **Third Party Servlet Support**: The HTTP task loads the JVM, but not the Domino servlet manager. This allows the use of third-party servlet managers such as IBM's WebSphere Application Server. Not supported on the AS/400 platform.
Servlet URL path	The path in a URL that signals Domino that the URL refers to a servlet. The default is /servlet.
Class path	A list of one or more paths that the servlet Manager class loader searches to find servlets and their dependent classes. This setting allows you to add additional paths. You may specify directories, JAR files, and ZIP files. Paths may be absolute or relative to the Domino data directory. The default is domino\servlet. **Tip:** If there is more than one path used, use the colon sign (":") to separate paths.
Session state tracking	**Enabled**: (default) The servlet manager periodically checks the user activity of all HttpSession instances. Sessions that have been idle for a given period of time are automatically terminated. The servlet manager calls the instance's HttpSession.invalidate() method to inform the servlet that the session is being terminated. **Disabled**: Sessions will not be checked for inactivity.
Idle session timeout	The number of minutes of user inactivity to wait before terminating a session. The default is 30 minutes.
Maximum active sessions	The number of simultaneous active sessions allowed. The default is 1,000 sessions. When this limit is reached, the sessions that have been idle the longest are terminated.

Setting	Options
Session persistence	**Enabled**: When the HTTP task exits, the servlet manager saves session data to a disk file called sessdata.ser in the Domino data directory. The session data will be reloaded when the HTTP task is restarted. Objects that the servlet has bound to sessions will also be saved if the objects implement the java.io.Serializable interface. **Disabled**: (default) All session data is discarded when the HTTP task exits.

In our examples, we have the following settings:

- **Java servlet support**: Domino Servlet Manager
- **Servlet URL path**: /servlet
- **Class path**: domino\servlet; domino\servlet\jt400.jar
- **Servlet file extensions**: empty
- **Idle session timeout**: 30 minutes
- **Maximum active sessions**: 1000
- **Session persistence**: Disabled

C.3.1 Setting properties for servlets

Special properties for individual servlets can be specified in a text file called *servlets.properties* located in the Domino data directory. The following properties can be specified:

- Alias
- Initialization arguments
- URL extension mapping
- Load at Servlet Manager startup

These properties are specified by directives in the servlets.properties file. The general syntax of a directive is:

```
servlet(s).<name>.<property>=<value(s)>
```

Directives are case-sensitive. The servlets.properties file can also contain blank lines and comment lines starting with the “#” character. The servlets.properties file is optional. The default properties for servlets are:

- No alias
- No initialization arguments
- No extension mapping
- Load servlets on demand

More information can be found in the Domino Designer help document *Running servlets in Domino*.

Note

The servlets.properties file *must* be in code page 819 or in code page 437. If it is not, it appears ignored.

The primary solution is to create the file using Windows Wordpad and save it directly in the Domino server data directory on the AS/400 system. This requires mapping a drive to the AS/400 server as already discussed in 11.1.2, “Certificate Authority setup” on page 269. Once the file is created, it can be edited using the Edit File (EDTF) command.

There is a secondary solution if you can't use the PC editor. If the file is first created using the Edit File (EDTF) command, the code page is 37. Rename the file and copy it to a new file with the proper name. On the copy, you have the option of setting the code page. For example, the following code allows you to get a servlets.properties file with the right code page:

```
CPY OBJ('/domino/domsvr5/servlets.propertiesx')
TOOBJ('/domino/domsvr5/servlets.properties') TOCODEPAGE(819)DTAFMT(*TEXT)
```

In our example, we have the following setting in the servlets.properties file:

```
servlet.OrderUpdateServlet.code = eboats.OrderUpdateServlet
servlet.OrderQueryServlet.code = eboats.OrderQueryServlet
servlets.startup= OrderUpdateServlet OrderQueryServlet
```

The last line above loads the servlets when the HTTP server starts.

Now, copy all the servlets and JavaBeans to the domino/servlet/eboats/ directory under the Domino server data directory. Restart your Domino HTTP server again. Type the following command on AS/400 command line to start the Domino console on the AS/400 system:

```
WRKDOMCSL dominoservername
```

On the console command line, type the following command to clear the caches:

```
Tell http ClearCaches
```

The following message is displayed when the command has run:

```
HTTP Web Server Caches Cleared
```

Type the following command to restart the HTTP server:

```
tell http restart
```

The following message is displayed when the command is finished:

```
HTTP Web Server restarting
Java Servlet Manager initialized
Addin: Agent printing: eboats.OrderUpdateServlet: init
Addin: Agent printing: eboats.OrderQueryServlet: init
HTTP Web Server restarted
```

Note

For performance reasons, it is important to create Java programs for all Java classes which you use regularly on the AS/400 system. This includes your own classes and all Java class libraries upon which you rely.

To create a Java program, you need to run the Create Java Program (CRTJVAPGM) command on the class, JAR file, or ZIP file. To check that a Java program already exists, use the Display Java Program (DSPJVAPGM) command.

More information about Java and Domino on the AS/400 system can be found in the "Java and Domino" chapter of the redbook *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345.

The deployment is now finished.

C.4 XML Java agent

This section describes all of the Java agents used in the sample application.

C.4.1 orderXML

This agent is stored in the eBoatsIntranet.nsf database. This agent is called when an intranet user clicks Java XML from the menu in the eboatsIntranet.nsf database. This agent reads records from the ORDERINFO table on the DB2 UDB for AS/400, and reforms it in the XML format. If the user uses Internet Explorer, the agent prints the XML data to the browser. If the user uses Netscape Communicator, the agent converts the XML and the XSL data to an HTML string, and prints the HTML string to the browser.

The sample code is shown in the following section.

C.4.1.1 JavaAgent

JavaAgent is the main entry of the orderXML agent. The code of the JavaAgent class is shown here:

```
// Sample agent to export Domino data in XML format.
// This agent creates generic XML from a Domino document, and optionally
// applies an XSLT stylesheet to it. Depending on the type of browser, the
// stylesheet is applied at the browser or at the server.
// The agent is controlled via arguments on the URL.
//
// The URL can specify a document in two ways:
//     ../ShowNote?OpenAgent&person=Joe+Person
// means to open the person document for "Joe Person" from names.nsf.
//     ../ShowNote?OpenAgent&db=something.nsf&unid=XXXXXXXXX
// means to open the document with unid XXXXXXXXX in database something.nsf.
//
// In either case, an argument of "&style=sheet.xsl" will use the stylesheet
// named "sheet.xsl" from this database to style the XML.
//
// This code uses the IBM XML4J parser and the LotusXSL processor available from
//   http://www.alphaworks.ibm.com
// Modify the JavaUserClasses notes.ini variable to put the xml4j.jar and lotusxsl.jar
// files in the path.

import java.io.*;
import java.util.*;
import lotus.domino.*;
import com.lotus.xsl.*;
```

```
import java.lang.*;
import java.sql.*;
import java.net.*;
import java.math.*;

public class JavaAgent extends AgentBase {
    private String orderXMLData = null;
    private String bpnum = null;
    private String bpNumber;
    private String orderDate;
    private String orderNumber;
    private java.math.BigDecimal prodCount;
    private String prodNumber;
    private String shipDate;
    private String shipStatus;
    private java.math.BigDecimal price;
    private final String SYSTEMNAME = "as20";
    private final String LIB = "EBOATS";
    private String statement = "SELECT ORDERNUMBER, ORDERDATE, BPNUMBER, SHIPDATE, PRODNUMBER,
PRODCOUNT, SHIPSTATUS, PRICE FROM EBOATS.ORDERINFO ";
//  private String statement = "call eboats.queryorder(?)";
    private java.sql.Connection dbConnection;
    private java.sql.PreparedStatement psQueryOrder;
//  private CallableStatement psQueryOrder;
    private ResultSet res;

    public void NotesMain() {

        try {

            // Parse the URL arguments
            UrlArguments args = new UrlArguments();
            args.fromSession(getSession());

                bpnum = args.getString("bpnumber");
            getOrderData();

            // Determine if the browser can render XML directly.
            boolean bXml = canBrowserDoXml(args);
            args.put("bXml", String.valueOf(bXml));// So we can see it in debug output

            // Two separate methods handle the two cases (browser can/can't
            // render XML), but both call the same method to generate the
            // XML itself.
            if (bXml) {
                DoXmlBrowser(args);
              }
            else {
                DoHtmlBrowser(args);
            }

        } catch(NotesException e) {
            getAgentOutput().println("<!-- ");
            getAgentOutput().println("Notes exception: " + e.text);
            e.printStackTrace(getAgentOutput());
            getAgentOutput().println("-->");
        } catch(Exception e) {
            getAgentOutput().println("<!-- ");
            e.printStackTrace(getAgentOutput());
            getAgentOutput().println("-->");
        }
    }

    public void connectToDB(String userid, String password) {
        // Create a properties object for JDBC connection
        Properties jdbcProperties = new Properties();

        // Set the properties for the JDBC connection
        jdbcProperties.put("user", userid);
        jdbcProperties.put("password", password);
        jdbcProperties.put("naming", "sql");
        jdbcProperties.put("errors", "full");
        jdbcProperties.put("date format", "iso");
        //jdbcProperties.put("extended dynamic", "true");
        //jdbcProperties.put("package", "SerTest");
        try {
//          System.out.println("before register driver");
```

```
          DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());
//Toolbox JDBC driver

//         System.out.println("before getConnection");
           dbConnection = DriverManager.getConnection("jdbc:as400://" + SYSTEMNAME + "/" +
LIB, jdbcProperties); //toolbox

       } catch (SQLException ex) {
//         System.out.println("connect failed");
           ex.printStackTrace();
           return;
       }
//     System.out.println("connected successfully");
       return;
   }

   public void closeConnect() {
       try{
           dbConnection.close();
       } catch ( java.sql.SQLException e){
       }
   }

    private String formatDate(String idate) {

      StringBuffer sbf = new StringBuffer(11);
      sbf.append(idate.substring(0,4));
      sbf.append(".");
      sbf.append(idate.substring(4,6));
      sbf.append(".");
      sbf.append(idate.substring(6));
      return sbf.toString();
   }

// This method get the data from the orderinfo database on AS/400 system and convert the data
to XML format.
    private void getOrderData()
          throws Exception
    {
    connectToDB("USERID", "PASSWORD");
         if (bpnum != null )
              statement = statement + " Where bpnumber = \'" + bpnum.trim() + "\' ";
         statement = statement + " order by bpnumber";
//System.out.print( statement);
       psQueryOrder = dbConnection.prepareCall(statement);
//     psQueryOrder.setString(1, iNum);

       res = psQueryOrder.executeQuery();
       StringBuffer tmpBuf = new StringBuffer(100000);
       tmpBuf.append("<ordercatalog>");
       while (res.next()){
           bpNumber = res.getString("BPNUMBER");
           orderDate = formatDate(res.getString("ORDERDATE"));
           orderNumber = res.getString("ORDERNUMBER");
           price = res.getBigDecimal("PRICE",2);
           prodCount = res.getBigDecimal("PRODCOUNT",2);
           prodNumber = res.getString("PRODNUMBER");
           shipDate = formatDate(res.getString("SHIPDATE"));
           shipStatus = res.getString("SHIPSTATUS");
           if (shipStatus.equals(new String("0")))
               shipStatus = "loaded";
           else if (shipStatus.equals(new String("1")))
               shipStatus = "shipping";
           else
               shipStatus = "received";
           tmpBuf.append("<order> ");
           tmpBuf.append("<ordernumber>"  + orderNumber + "</ordernumber> ");
           tmpBuf.append("<orderdate>" + orderDate + "</orderdate> ");
           tmpBuf.append("<bpnumber>" + bpNumber + "</bpnumber> ");
           tmpBuf.append("<shipdate>" + shipDate + "</shipdate> ");
           tmpBuf.append("<prodnumber>" + prodNumber + "</prodnumber> ");
           tmpBuf.append("<prodcount>" + prodCount.toString() + "</prodcount> ");
           tmpBuf.append("<shipstatus>" + shipStatus + "</shipstatus> ");
           tmpBuf.append("<price>" + price.toString() + "</price>");
           tmpBuf.append("</order> ");
           }
           tmpBuf.append("</ordercatalog>");
           orderXMLData = tmpBuf.toString();
```

```
/*           orderXMLData = " <ordercatalog> <order>
<ordernumber>100001</ordernumber><orderdate>20000123</orderdate><bpnumber>1004</bpnumber>"
                                        +
"<shipdate>20001201</shipdate><prodnumber>100004</prodnumber><prodcount>12.4</prodcount>"
                                        +
"<shipstatus>loaded</shipstatus><price>134.5</price></order></ordercatalog>";
*/
        closeConnect();
     }

    // This method handles the case of browsers that can render XML directly.
    // It gets the XML from DoXmlWork, and sends it directly to the browser with
    // the proper declarations to have it interpreted as XML and styled properly.

    private void DoXmlBrowser(UrlArguments args)
        throws Exception
    {
        // The URL might specify a stylesheet
        String styleName = "ord.xsl";

        PrintWriter out = getAgentOutput();

        // We'll be writing XML to the agent's stdout.
        XmlWriter xml = new XmlWriter(out);

        xml.WriteContentTypeHeader();
        xml.WriteXmlDeclaration();

        // If we had a stylesheet, write the processing instruction that
        // references it.
        if (styleName != null) {
            xml.UseStylesheet("styles/" + styleName, "text/xsl");
        }

        // Do the main chunk of work: the same in both cases.
        DoXmlWork(xml);
    }

    // This method handles the case of browsers that can't render XML directly.
    // It gets the XML from DoXmlWork, and then processes it with the LotusXML
    // engine to convert it into something the browser can handle.

    private void DoHtmlBrowser(UrlArguments args)
        throws Exception
    {
        // The URL might specify a stylesheet
           String styleName = "ord.xsl";
        PrintWriter out = getAgentOutput();

        // We'll be writing HTML to the agent's stdout. The XML will be written
        // to a character buffer, and then processed by LotusXSL.
        CharArrayWriter xmlbuf = new CharArrayWriter();
        XmlWriter xml = new XmlWriter(xmlbuf);
        xml.WriteXmlDeclaration();

        // Do the main chunk of work: the same in both cases.
        DoXmlWork(xml);
        // Get the xsl transform spec.
        if (styleName != null) {
            final String oldURI = "xmlns:xsl=\"http://www.w3.org/TR/WD-xsl\"";
            final String newURI = "xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\"
version=\"1.0\"";

            // Because the standards are in flux, different implementations don't
            // agree on all the details. IE5 and LotusXSL 0.17.1 don't agree on
            // the URI that indicates the xsl namespace.
            // Need to change:
            //     http://www.w3.org/TR/WD-xsl
            // to:
            //     http://www.w3.org/xsl/transform/1.0
            String xsl = replaceSubstring(getStylesheet(styleName), oldURI, newURI);

            // Run it through the XSL processor.
            //XSLProcessor xslEngine = new
XSLProcessor("com.lotus.xml.xml4j2dom.XML4JLiaison4dom");
             XSLProcessor xslEngine = new XSLProcessor(new com.lotus.xml.xml4j.ProcessXSL());

            xslEngine.process(
```

```
                new CharArrayReader(xmlbuf.toCharArray()),
                new StringReader(xsl),
                "",
                out);
        }
    }

    // This method does the work of finding the Domino document indicated in the
    // URL arguments, and converting it to XML.

    private void DoXmlWork(XmlWriter xml)
        throws Exception
    {
           xml.WriteString( orderXMLData );
        // A "person" arg specifies a document via the Directory
    }

    // Find a person document in the Domino Directory.


    // Determine if the browser can handle XML natively. Right now, this
    // simply tests for whether it is IE 5 or not.

    private boolean canBrowserDoXml(UrlArguments args)
    {
        String userAgent = args.getString("HTTP_USER_AGENT");
        if (userAgent != null && userAgent.indexOf("MSIE 5.") > 0) {
            return true;
        }
        return false;
    }

    // Open one of our stylesheet documents and retrieve the text of the stylesheet.
    // This is used only in the case of a non-XML browser, so that we can feed the
    // stylesheet to the LotusXSL processor. In the case of an XML browser, the
    // browser requests the stylesheet from the web server directly.

    private String getStylesheet(String style)
        throws lotus.domino.NotesException
    {
        Database db = getSession().getAgentContext().getCurrentDatabase();
        View styles = db.getView("Styles");
        Document stylesheet = styles.getDocumentByKey(style);
        if (stylesheet == null)
            return null;

        return stylesheet.getItemValueString("HTML");
    }

    // A simple method to replace a substring with another substring.

    private String replaceSubstring(String s, String oldSub, String newSub)
    {
        int iSub = s.indexOf(oldSub);
        if (iSub < 0)
            return s;
        return s.substring(0, iSub) + newSub + s.substring(iSub + oldSub.length());
    }
}

// end of class JavaAgent
```

C.4.1.2 URLArguments class

Apart from the JavaAgent class, the orderXML agent contains two other classes which are called inside the JaveAgent class logic. The first one is URLArguments. It is dedicated to parse arguments from a URL and can be used in any Java agent. The following is the code of the URLArguments class:

```
// This class parses arguments from a URL.
// A URL like this:
//      ...?OpenAgent&foo=bar&baz&quux=wombat
// will result in a hashtable populated like this:
//  "foo"->"bar"
//  "baz"->""
//  "quux"->"wombat"
```

```
//
// In addition, the CGI variables are populated into the arg table, so you
// can retrieve HTTP_USER_AGENT, etc, from the table directly.
//
// This is a general-purpose class that can be used in any agent.

import lotus.domino.*;
import java.util.*;

public class UrlArguments extends Hashtable {
    public UrlArguments() {}

    public void fromString(String args) {
        StringTokenizer toks = new StringTokenizer(args, "&=", true);
        String tok = toks.nextToken();// first token is command: skip it.
        String key = null;
        boolean bKeyNext = true;

        while (toks.hasMoreTokens()) {
            tok = toks.nextToken();
            if (tok.equals("&")) {
                if (bKeyNext && (key != null)) {
                    // Null value
                    put(key, "");
                }
                key = null;
                bKeyNext = true;
            }
            else if (tok.equals("=")) {
                bKeyNext = false;
            }
            else {
                if (bKeyNext) {
                    key = new String(tok);
                }
                else {
                    if (key != null) {
                        put(key, tok);
                    }
                    key = null;
                }
            }
        }

        // We have to special-case a valueless argument at the end.
        if (bKeyNext && (key != null)) {
            // Null value
            put(key, "");
        }
    }

    public void fromSession(Session session)
        throws lotus.domino.NotesException
    {
        AgentContext agentContext = session.getAgentContext();
        Document doc = agentContext.getDocumentContext();

        // First add all of the CGI variables. They are in the document context.
        Vector items = doc.getItems();
        for (int iItem = 0; iItem < items.size(); iItem++) {
            Item item = (Item)items.elementAt(iItem);
            put(item.getName(), item.getText());
        }

        // Now parse the URL arguments themselves.
        fromString(doc.getItemValue("QUERY_STRING_DECODED").elementAt(0).toString());
    }

    // A simple accessor to make casts unnecessary.
    public String getString(String key) {
        return (String) get(key);
    }
}

// end of class UrlArguments
```

C.4.1.3 XmlWriter class

The XmlWriter class is the second additional class in the orderXML agent. It is also called inside the JavaAgent class logic. Its function is to format XML data. The following is the code of the XmlWriter class:

```
// XmlWriter handles the job of writing data as XML to a java.io.Writer object.
//
// This class handles both the low-level tasks of writing character data properly
// as XML, up through Domino datetimes, up to entire Domino documents.

import java.io.*;
import java.lang.*;
import java.text.*;
import java.util.*;
import lotus.domino.*;

public class XmlWriter {
    // This is the base Writer that we were told to write to.
    private Writer w;

    // To construct one of these, just tell us where you want the data to go to.

    public XmlWriter(Writer writer) {
        w = writer;
    }

    // Write out the content type HTTP header that tells the browser that XML is coming.

    public void WriteContentTypeHeader()
        throws java.io.IOException
    {
        w.write("Content-Type: text/xml\n");
    }

    // Write out the XML declaration that should start all XML streams.

    public void WriteXmlDeclaration()
        throws java.io.IOException
    {
        w.write("<?xml version='1.0' encoding='UTF-8'?>\n");
    }

    // Write a processing instruction that links the XML to a stylesheet.

    public void UseStylesheet(String href, String type)
        throws java.io.IOException
    {
        w.write("<?xml:stylesheet type='" + type + "' href='" + href + "'?>\n");
    }

    // Write a simple comment. The text should not contain "--".

    public void Comment(String text)
        throws java.io.IOException
    {
        w.write("<!-- " + text + " -->\n");
    }

     // Write a string as plain text
     public void WriteString( String s)
          throws java.io.IOException
     {
         w.write( s );
          w.flush();
     }
    // This function handles quoting of special characters in XML output for us.
    // XML is not allowed to contain < > & ' or " except where they are used in
    // the tag syntax, so any occurence in text content must be as entities.

    public void WriteChars(String s)
        throws java.io.IOException
    {
        StringTokenizer chunks = new StringTokenizer(s, "<>&'\"", true);
        String chunk;

        while (chunks.hasMoreTokens()) {
```

```
        chunk = chunks.nextToken();
        if (chunk.length() == 1) {
            switch (chunk.charAt(0)) {
                case '<': w.write("&lt;"); break;
                case '>':  w.write("&gt;"); break;
                case '&':  w.write("&amp;"); break;
                case '\'': w.write("&apos;"); break;
                case '"':  w.write("&quot;"); break;
                default:   w.write(chunk); break;
            }
        }
        else {
            w.write(chunk);
        }
    }
}

// This function writes a DateTime in ISO8601 format.

public void WriteDateTime(DateTime dt)
    throws java.io.IOException, lotus.domino.NotesException
{
    SimpleDateFormat iso8601
        = new SimpleDateFormat("yyyyMMdd'T'hhmmss','SSS");
    int zone = dt.getTimeZone();
    String zonePrefix = "";
    if (zone < -9) {
        zonePrefix = "-";
        zone = -zone;
    }
    else if (zone < 0) {
        zonePrefix = "-0";
        zone = -zone;
    }
    else if (zone < 10) {
        zonePrefix = "+0";
    }
    else {
        zonePrefix = "+";
    }

    String dateString;
    try {
        dateString =
            iso8601.format(dt.toJavaDate()) +
            zonePrefix +
            Integer.toString(zone);
    }
    catch (Exception e) {
        dateString = "";
    }

    w.write(dateString);
}

// Write a Domino document out in a generic XML dtd.
// If bEmptyItems is false, then only items with content will be output.

public void WriteDocument(Document doc, boolean bEmptyItems)
    throws java.io.IOException, lotus.domino.NotesException
{
    w.write("<note>\n");

    Vector items = doc.getItems();
    for (int iItem = 0; iItem < items.size(); iItem++) {
        Item item = (Item)items.elementAt(iItem);

        // We can only handle simple types.
        boolean bXmlable = false;

        switch (item.getType()) {
        case Item.TEXT:
        case Item.NUMBERS:
        case Item.DATETIMES:
        case Item.NAMES:
        case Item.READERS:
        case Item.AUTHORS:
            bXmlable = true;
```

```
            }

            Vector vValues = null;

            if (bXmlable) {
                // Get the values
                vValues = item.getValues();
            }

            if (vValues == null && !bEmptyItems) {
                continue;
            }

            w.write("<item name='" + item.getName() + "'");

            if (vValues == null) {
                // An empty item value
                w.write("/>");
                continue;
            }

            // The item has content we can handle
            w.write(">");

            int nValues = vValues.size();
            if (nValues > 1) {
                w.write("<list>");
            }

            switch (item.getType()) {
            case Item.TEXT:
            case Item.NAMES:
            case Item.READERS:
            case Item.AUTHORS:
                for (int iValue = 0; iValue < nValues; iValue++) {
                    w.write("<text>");
                    WriteChars((String) vValues.elementAt(iValue));
                    w.write("</text>");
                }
                break;

            case Item.NUMBERS:
                for (int iValue = 0; iValue < nValues; iValue++) {
                    w.write("<number>");
                    WriteChars(vValues.elementAt(iValue).toString());
                    w.write("</number>");
                }
                break;

            case Item.DATETIMES:
                for (int iValue = 0; iValue < nValues; iValue++) {
                    w.write("<datetime>");
                    WriteDateTime((DateTime) vValues.elementAt(iValue));
                    w.write("</datetime>");
                }
                break;

            default:
                w.write("<other/>");
                break;
            }

            if (nValues > 1) {
                w.write("</list>");
            }

            w.write("</item>\n");
        }

        w.write("</note>\n");
        w.flush();
    }
}

// end of class XmlWriter
```

C.4.2 Agent requirement

To compile this agent on your PC, you should have five ".jar" files in your Notes client NOTES.INI file. For example, place the following line into your NOTES.INI on your PC:

```
JavaUserClasses=e:\XML4J_3_0_0EA3\xml4j.jar;e:\lotusxsl_0_19_2\
lotusxsl.jar;e:\XML4J_3_0_0EA3\xerces.jar;e:\jt400\lib\jt400.jar;
```

The .jar files should be in the specific path on your hard disk.

To run this agent on the Domino server 5.0.2, you should have the four .jar files in your server NOTES.INI file. For example, place the following line into your NOTES.INI file on your server:

```
JavaUserClasses=/lotus/domino/Domsvr5/domino/Java/xml4j.jar:/lotus/domino/
Domsvr5/domino/Java/lotusxsl.jar:/lotus/domino/Domsvr5/domino/Java/
xerces.jar:/lotus/domino/Domsvr5/domino/Java/NCSO.jar:/lotus/domino/Domsvr5/
domino/Java/jt400.jar:
```

Make sure that you are using a colon to separate the .jar files. Make sure that the .jar files are in the correct directory and have the AS/400 user profile QNOTES as their owner.

Note

For performance reasons, it is important to create Java programs for all Java classes that you use regularly on the AS/400 system. This includes your own classes and all Java class libraries upon which you rely.

To create a Java program, you need to run the Create Java Program (`CRTJVAPGM`) command on the class, JAR file, or ZIP file. To check that a Java program already exists, use the Display Java Program (`DSPJVAPGM`) command.

More information about Java and Domino on the AS/400 system can be found in the "Java and Domino" chapter of the redbook *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345.

C.5 Servlet forcing SSL usage

The following servlet is not part of the eBoats International sample application. The code is shown here to provide an example of how to force SSL usage in a servlet.

The code uses standard JSDK APIs:

- The getScheme() method returns the scheme of the URL used in this request, for example "http", "https", or "ftp".
- The getServerName() method returns the host name of the server that received the request.
- The getRequestURI() method gets, from the first line of the HTTP request, the part of this request's URI that is to the left of any query string.
- The getQueryString() method gets any query string that is part of the HTTP request URI.

- The sendRedirect(String location) method sends a temporary redirect response to the client using the specified redirect location URL.

The sample servlet code is shown here:

```
public void doGet(HttpServletRequest req,
                  HttpServletResponse response)
          throws ServletException, IOException
   {
       try
       {
       NotesThread.sinitThread();
       if (req.getScheme().equalsIgnoreCase("https")) {
           // Process request
           response.setContentType("text/html");
           ServletOutputStream out = response.getOutputStream();
           out.println("<html>");
           out.println("<head><title>HelloWorld with forced SSL</title></head>");
           out.println("<body>");
           out.println("<h1>Hello World with SSL</h1>");
           Session s = Session.newInstance();
           out.println("<PRE>Current Domino user name (server name) is: " + s.getUserName());
           out.println("Current Domino version: " + s.getNotesVersion());
           out.println("Current Domino platform: " + s.getPlatform());
           DateTime dt = s.createDateTime("Today");
           dt.setNow();
           out.println("<font color=green>Date is now: " + dt.getDateOnly() + "</font>");
           out.println("<font color=red>Time is now: " + dt.getTimeOnly() + "</font></PRE>");
           out.println("</body></html>");
            }
        else {
              String sslUrl = "https://" + req.getServerName() + req.getRequestURI();
              if (req.getQueryString() != null)
                 sslUrl += "?" + req.getQueryString();
              response.sendRedirect(sslUrl);
            }

       }  catch (Exception e) {
          e.printStackTrace();
       }

       finally { NotesThread.stermThread(); }
   }
```

Figure 265 shows the output of the servlet, as displayed in the Web browser.

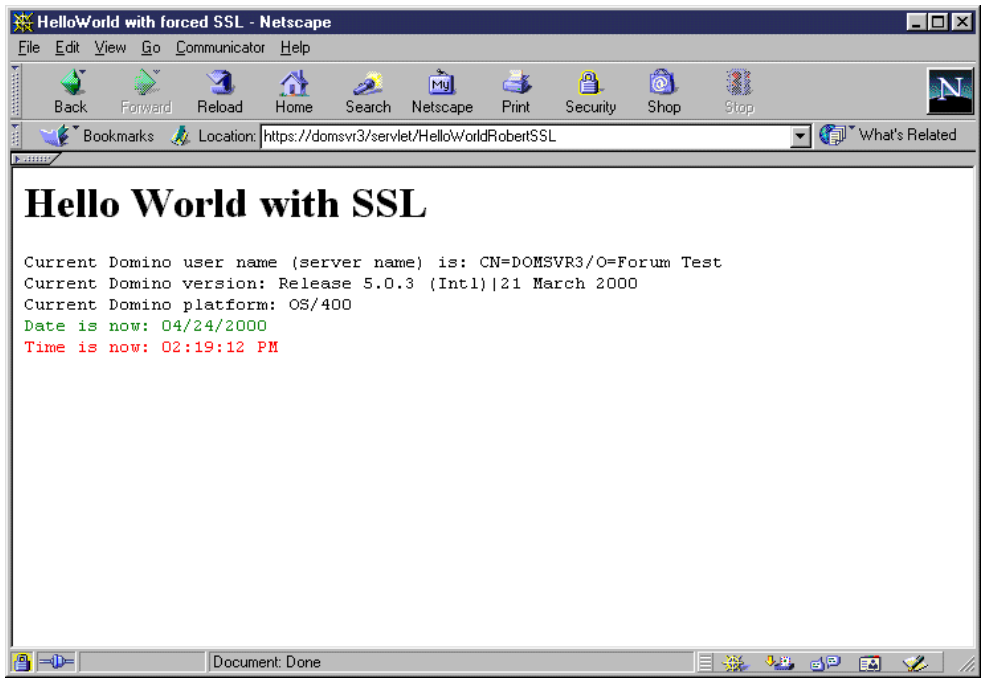


Figure 265. Using a servlet forcing SSL usage

Appendix D. Other sample code

This appendix contains other coding examples demonstrated in the sample database. Each section describes different code examples.

D.1 @DbCommand

This is the code behind the Check Using @DbCommand button on the InventoryQty form in the eboatsB2B.nsf database. This code does a lookup to the PRODUCTS DB2 UDB for AS/400 table and returns the current inventory quantity for the selected model:

```
mdl := @DbLookup( "" : "NoCache" ; "" : "eboatsInternet.nsf"  ; "(AllProducts)" ; Model ; 3 );
qryString := "SELECT PRODAVAILABLE FROM EBOATS.PRODUCTS  WHERE PRODNUMBER=\'"  + mdl + "\'" ;

res := @DbCommand( "ODBC" : "NoCache" ; "AS20"  ; "USERID" ; "PASSWORD" ; qryString );
@SetField( "Results" ; res )
```

D.2 Stored procedure queryOrder

This procedure is stored on the AS/400 system in the eboats library and is called by the OrderInfoQuery JavaBean. This stored procedure executes an SQL query from the ORDERINFO DB2 UDB for AS/400 table:

```
create procedure eboats/queryOrder ( in inum char(4))
result set 1
language sql
begin
    DECLARE c1 CURSOR FOR
        SELECT ORDERNUMBER, ORDERDATE, BPNUMBER, SHIPDATE,
               PRODNUMBER, PRODCOUNT, SHIPSTATUS, PRICE
        FROM eboats/orderinfo WHERE BPNUMBER = inum;
    open c1;
    set result sets cursor c1;
end
```

D.3 orderCatalog.xsl page

This page is contained in the eboatsIntranet.nsf database. It contains the XML formatting code used to display order status:

```
<?xml version="1.0"?>
      <xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
     <xsl:template match="/" >
       <TABLE STYLE="border:3px inset black">
         <TR STYLE="font-size:10pt; background-color: lightgrey;
         font-family:Verdana; font-weight:bold">
         <TD>Order Number</TD>
         <TD>Order Date</TD>
         <TD>BP Number</TD>
         <TD>Ship Date</TD>
         <TD>Product Number</TD>
         <TD>Quantity</TD>
         <TD>Ship Status</TD>
         <TD>Price</TD>
       </TR>
       <xsl:for-each select="ordercatalog/order" order-by="bpnumber">
         <TR STYLE="font-family:Verdana; font-size:10pt; padding:0px 6px">
          <TD><xsl:value-of select="ordernumber" /></TD>
          <TD><xsl:value-of select="orderdate" /></TD>
          <TD><xsl:value-of select="bpnumber" /></TD>
          <TD><xsl:value-of select="shipdate" /></TD>
          <TD><xsl:value-of select="prodnumber" /></TD>
          <TD><xsl:value-of select="prodcount" /></TD>
          <TD><xsl:value-of select="shipstatus" /></TD>
          <TD STYLE="background-color:lightgrey">$
```

```
          <xsl:value-of select="price" /></TD>
          </TR>
          </xsl:for-each>
        </TABLE>
      </xsl:template>
</xsl:stylesheet>
```

Appendix E. Integration with WebSphere on the AS/400 system

As explained in 1.1.4, “Lotus and IBM strategy” on page 5, the Domino and WebSphere application servers both offer strong Web application server capabilities that do not compete but complement each other. This appendix presents the WebSphere Application Servers family and how WebSphere and Domino for AS/400 can coexist and be integrated on the AS/400 system. More integration functions should become available with Domino for AS/400 and WebSphere in the next releases.

E.1 WebSphere Application Servers family

The IBM WebSphere Application Servers provide the runtime environment for your Web applications and an administration interface for managing your Web applications, Web resources, and Web users. There are a number of Editions of the WebSphere Application Server that you can choose from depending upon your needs. Each of them is designed with a particular environment in mind. These range from simple Web application environments to very complex e-business environments. The WebSphere Application Servers support industry standard Web and Java programming models including HTML, Java servlets, JavaServer Pages (JSPs), and Enterprise JavaBeans (EJBs).

There are three editions of WebSphere Application Servers:

- WebSphere Application Server Standard Edition
- WebSphere Application Server Advanced Edition
- WebSphere Application Server Enterprise Edition

E.1.1 WebSphere Application Server Standard Edition

The IBM WebSphere Application Server Standard Edition is a robust deployment environment for e-business applications. Its components let you build and deploy personalized, dynamic Web content quickly and easily. Using open Java-based technologies and application programming interfaces (APIs) as well as the latest Extensible Markup Language (XML) technologies, the Standard Edition lets you leverage your existing resources, shorten development cycles, and ease your administrative burden.

The Standard Edition extends the value and versatility of your Web server with:

- Full Java servlet, JavaBeans, JavaServer Pages (JSP), Extensible Markup Language (XML), and Extensible Stylesheet Language (XSL) support, including support for applications built with the Java servlet specification
- The Java servlet 2.1 API, for easy, automatic creation of users sessions and state information
- Database pooling for dynamic access to databases with Java Database Connectivity (JDBC) access, including DB2 Universal Database, Oracle, and Microsoft SQL Server
- Improved integration with IBM VisualAge for Java
- A site analysis tool that lets you gauge your Web site's traffic volume, identify its traffic sources, and manage your site's integrity

- Extended tagging support through JSP specification 1.0 for queries and connection management
- Extensive monitoring tools that allow you to watch your site's servlets and sessions with nearly real-time capability
- Tivoli Read modules that can be managed by Tivoli-based tools

E.1.1.1 XML management

The Standard Edition includes support for Extensible Markup Language (XML) and Extensible Style Language (XSL) technologies to improve data definition and sharing, allowing data presentation to be separated from your actual data. The Standard Edition includes tools that support:

- An XML and XSL parser utilizing the latest W3C XML 1.0 specifications
- Dynamic HTML generation based on XML DOM Level 1 recommendations
- An XML DTD library for local validation of a document's structure
- An XSL processor to transform your XML data into formatted HTML

Additional tools enable you to tailor data to specific devices, including initial support for Wireless Markup Language (WML).

E.1.1.2 Security management

The Standard Edition contains a Lightweight Directory Access Protocol (LDAP) client for connection to an LDAP server. LDAP is an open, vendor-neutral standard that provides an extendable architecture for centralized storage and management of information.

The secure access control lists allow you to protect resources running within the application server. In addition to setting up security at the user and group levels, control and policies can be established for specific calls or methods within the applications themselves, using third-party authentication techniques. This provides much greater depth of control and protection within the server deployment environments.

E.1.1.3 Site analysis

The Standard Edition includes a package of site usage and analysis tools to help you enhance and improve your Web site's content and performance. The site analysis component lets you gauge your Web site's traffic volume (hits and visits), identify its traffic sources (domains, subdomains, and referrers), and manage your site's integrity (link verification and site conformance).

From the server-side, analyzers transform the raw server log data into valuable information which is stored in a DB2 Universal Database that is included. From the client, you can schedule the site analysis component to automatically generate static or dynamic HTML reports from analysis results, or you can schedule analysis tasks to run at specific times or time intervals. When the analysis is complete, you can generate customized reports and print the data or view it immediately from the client.

The features of the site analysis component include:

- Pre-defined, ready-to-use reports
- Ability to customize reports

- Detailed analysis of Web content integrity, site performance, and usage statistics
- Flexible client/server architecture for use within a team environment
- A database to provide scalability and trending with historical data

E.1.1.4 Server Manager

The Application Server Manager tools provide a single site solution for integrating the basic management of applications and their data with the overall management of the Web site.

The System Management tools provide support for:

- User and group management
- Authorization management
- Web application deployment
- Server operations

E.1.1.5 Web Server

WebSphere Application Server Standard Edition integrates with the AS/400 HTTP Server via two plug-in extensions that augment the HTTP server. The Services plug-in enables the Web server to use the same security, naming, and trace services that are used by the Standard application server runtime. The NCF plug-in enables the Web server to route uniform resource locators (URLs) that match administrator-defined filters to the Web application server. These URLs represent either servlet or JSP requests.

E.1.2 WebSphere Application Server Advanced Edition

IBM WebSphere Application Server Advanced Edition V3.0 is a high-performance Enterprise JavaBeans (EJB) server that implements EJB components that incorporate business logic. It supports multiple platforms, databases and transaction systems, providing Java-based gateway and EJB connectivity.

The Advanced Edition has the capabilities of the Standard Edition including:

- Site usage logging and analysis tools
- Java Virtual Machine (JVM) pooling, clustering, and connection pooling for performance, scaling, and reliability
- Open support for Java IDEs
- Lightweight Directory Access Protocol (LDAP) client/server support
- Tivoli Ready modules that can be managed by Tivoli-based tools

The Advanced Edition also includes:

- Workload manager for EJBs, providing EJB server clustering and workload distribution across servers
- SecureWay Directory Server as an optional directory supporting LDAP
- Container-managed and bean-managed persistence for EJBs using IBM DB2 server or, optionally, an Oracle server
- Granular EJB-level distributed security with third-party authentication and secure association services across multiple EJB servers

- A comprehensive Internet Inter-ORB Protocol (IIOP)-based administrative client for managing the execution of EJBs
- Support for EJB-based connectivity, Remote Method Invocation and IIOP (RMI/IIOP) communications with a new Java-based Object Request Broker (ORB) and Java Database Connectivity (JDBC)-XA connectivity for distributed, coordinated DB2 transactions
- All function and support for servlets, JavaBeans, and JavaServer Pages from Standard Edition

E.1.2.1 Performance and scalability

The Advanced Edition includes application-level workload management and clustering, with enhanced container deployment environment services for EJBs, servlets and JSPs. Improved transaction management intelligently deploys and executes across multiple applications and components, therefore optimizing object management and performance.

E.1.2.2 Transaction management

The Advanced Edition includes deployment and management capabilities for Java applications and EJB components that allow powerful interactions with enterprise databases, transaction processing systems and other applications. A single, robust server offers better control, more flexibility and better serviceability for the deployment and management of JavaBeans, Java servlets, JSPs, and applications built to the EJB specification for execution and transaction management.

Container management and persistent storage with DB2 Universal Database help provide a high-performance transactional environment using servlets and EJBs. The Advanced Edition also supports Oracle database servers for persistent storage of transactional EJBs. Thus, an Oracle database server can be swapped in and used with or instead of the DB2 Universal Database server.

E.1.2.3 Protocol and application adapter support

The Advanced Edition contains enhanced protocol and application programming interface (API) support. An integrated Java-based ORB within the EJB server delivers function for monitoring and controlling containers used within the server. Internet Inter-ORB Protocol (IIOP) support and Remote Method Invocation (RMI) over IIOP support improve object-oriented connections to Common Object Request Broker Architecture (CORBA)-compliant objects, ORBs, and applications. Support for the latest JDBC API provides flexible and robust connectivity to remote relational database systems supporting distributed database transactions, optimizing object management and performance.

E.1.2.4 Security management

The Advanced Edition contains support for LDAP-based user registries. LDAP is an open, vendor-neutral standard that provides an extendable architecture for centralized storage and management of information. LDAP-based directories give you the manageability and security you want to ensure that your information can be shared but still protected.

In addition to setting up security at the user and group levels, control and policies can be established for specific calls or methods within the applications themselves, using third-party authentication techniques. This offers a much

greater depth of control and protection within the server deployment environments.

E.1.2.5 Site analysis

From the server-side, analyzers transform the raw server log data into valuable information, which is stored in a DB2 Universal Database. From the client, you can schedule the site analysis component to automatically generate static or dynamic HTML reports from analysis results, or you can schedule analysis tasks to run at specific times or time intervals. When the analysis is complete, you can generate customized reports and print the data or view it immediately from the client.

The features of the site analyzer component include:

- Pre-defined, ready-to-use reports
- Ability to customize reports
- Detailed analysis of Web content integrity, site performance and usage statistics
- Flexible client/server architecture for use within a team environment
- A database to provide scalability and trending with historical data

E.1.3 WebSphere Application Server Enterprise Edition

IBM WebSphere Application Server Enterprise Edition enables full e-business transactions over the Web. Using such open standards-based technologies as interoperable CORBA and Enterprise JavaBeans (EJBs), Enterprise Edition provides comprehensive, high quality middleware runtime services for distributed component applications. It also contains the industry's most complete support for integrating existing IT applications and resources for reuse on the Web.

The Enterprise Edition offers comprehensive support by providing:

- The ability to compose new business applications that provide transactional commit and recovery (including full two-phase commit) across disparate backend systems, such as XA-compliant relational database management systems (RDBMSs) and transaction processing (TP) monitor applications using IBM CICS and IBM IMS systems
- A full complement of EJB-based interfaces to existing resource managers (databases, messaging and queuing systems, TP monitors and other vendor Enterprise Resource Planning (ERP) systems) through the IBM common connectivity approach
- Tunable, queryable, cacheable, performance-enhancing technology for interfacing with existing backend datastores

In addition, the Enterprise Edition offers the highest levels of scalability, performance availability, and configuration.

Note

Since the AS/400 system does not have a legacy of either TXSeries or Component Broker applications and since it provides transactional capabilities in the operating system, Enterprise application server is not supported on the AS/400 system. However, either the Standard or Advanced application server running on the AS/400 system can coexist and work with an Enterprise application server in the same network.

E.2 Coexisting on the same AS/400 system

Lotus Domino for AS/400 and WebSphere Application Server are licensed programs that can both be installed on the AS/400 system without any conflict. They can coexist on the same AS/400 system in a number of different scenarios.

E.2.1 Coexisting on a single system configuration

Domino and WebSphere can both be installed and run in a single system configuration on the AS/400 system. This coexistence is shown in Figure 266.

In this scenario, WebSphere is installed on the native IBM HTTP server that would handle all HTTP requests for WebSphere. Also, Domino can be configured with its normal HTTP services being provided by the HTTP task from within the Domino application server.

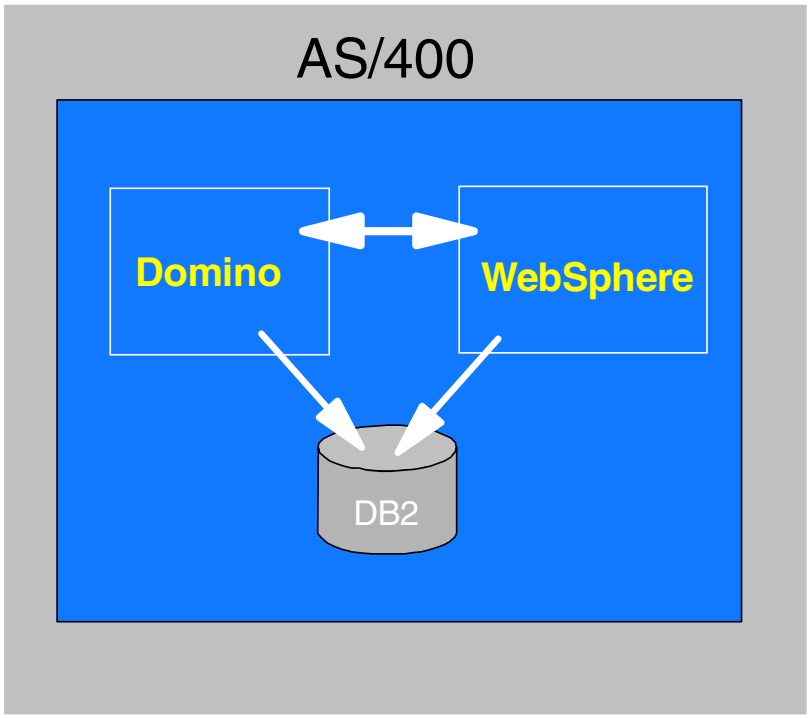


Figure 266. Coexisting on a single system configuration on the AS/400 system

E.2.1.1 Coexisting on the same logical partition

Domino and WebSphere can also be installed and run in a secondary logical partition on the AS/400 system. This coexistence is shown in Figure 267.

In this scenario, you would need at least a two-way AS/400 system with one primary logical partition and one secondary logical partition. On the primary partition, you may decide to have your DB2 database installed and you may also install other applications that you prefer to have in the primary logical partition. You then install Domino and WebSphere on the secondary logical partition.

The advantage of using a secondary logical partition for Domino and WebSphere is to enable you to Initial Program Load (IPL) the secondary logical partition in case of a problem with either products without having to take down the entire system. You also have the advantage of having two server environments as opposed to a single one. This means that you can separate your e-business environment from your production environment.

The drawback to this scenario is the extra resources needed to run logical partitions. Another drawback is that of accessing resources that reside on the primary partition. There would be a slight overhead due to the fact that the logical partitions are treated as two separate servers.

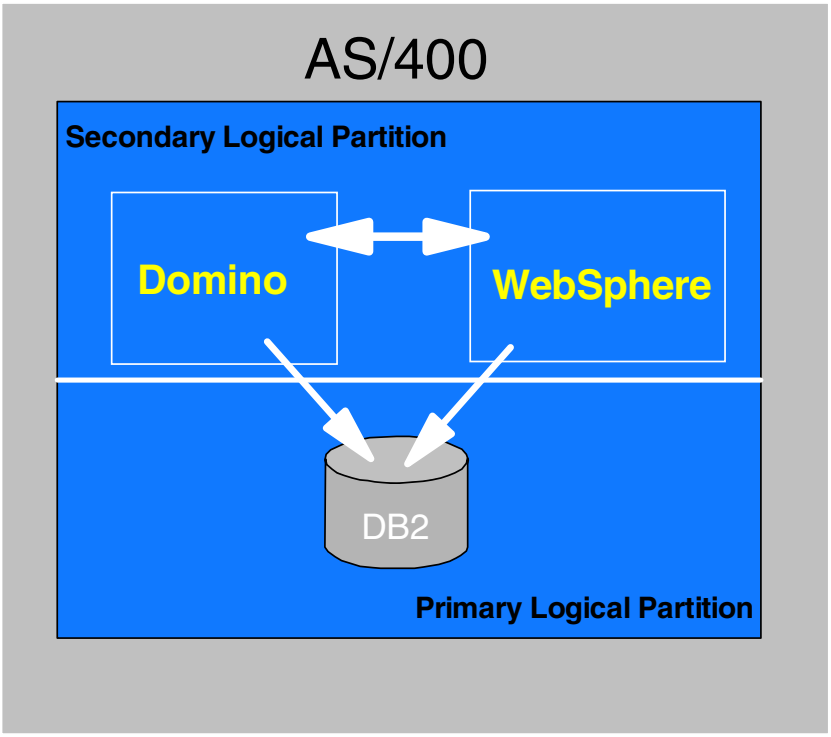


Figure 267. Coexisting on the same logical partition on the same AS/400 system

E.2.1.2 Coexisting on different logical partitions

Domino and WebSphere can both be installed and run within a secondary logical partition on the AS/400 system. This coexistence is shown in Figure 268 on page 382.

In this scenario, you would need at least a 4-way AS/400 system with one primary logical partition and two secondary logical partitions. On the primary partition, you may decide to have your DB2 database installed and you may also install other applications that you prefer to have in the primary logical partition. You then install Domino on one of the secondary logical partition and WebSphere on the other partition.

The advantage of using this scenario is the increased performance. However, this increase in performance would only be realized if the logical partitions are big enough and have the required resources to perform. This scenario also enables you to IPL the secondary logical partitions in case of a problem with either logical partitions without having to take down the entire system. You also have the advantage of having three server environments, as opposed to a single one or a double. This means that you can further separate your e-business components into individual systems and also from your production environment.

The drawback to this scenario is the extra resources needed to run logical partitions. Another drawback is that of accessing resources that reside on the other partitions. There would be some overheads, due to the fact that the logical partitions are treated as separate servers.

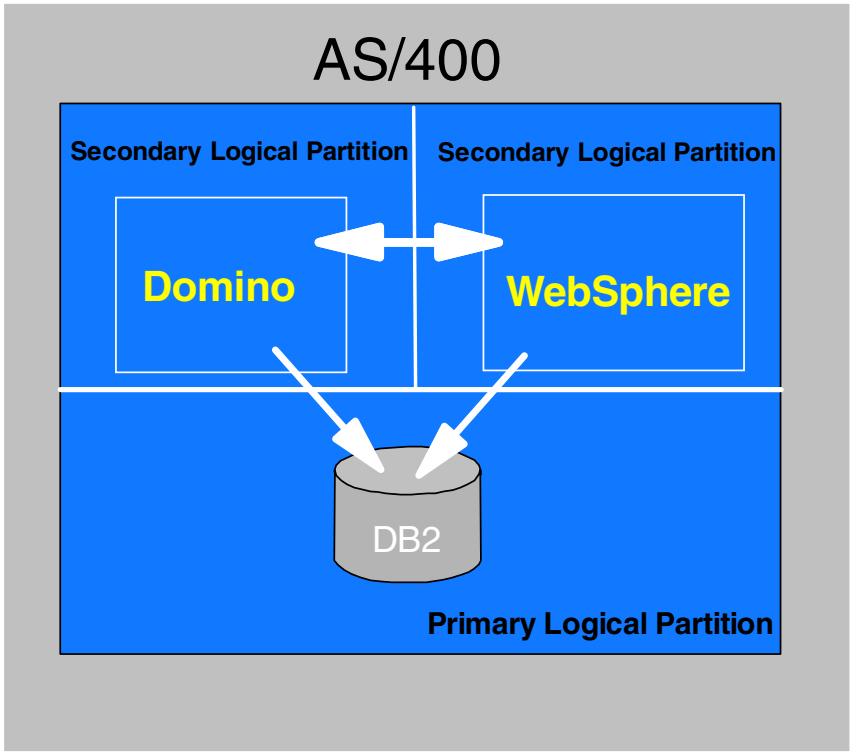


Figure 268. Coexisting on different logical partitions on the same AS/400 system

E.2.2 Sharing the same security environment

Sharing the same security environment between Domino and WebSphere gives you the possibility of administering all your security issues from a single point of view. In the AS/400 system, all resources are protected, and all the users of these resources must have user profiles in the AS/400 system, before they can have access. All the user profiles can be imported into a directory server. This can be used as base for authentication and authorization from both Domino and WebSphere.

Another way of sharing the same security environment is by using the Domino Directory. Both Domino and WebSphere can access the Domino directory so that could also be used. This scenario is discussed in E.3.2, “Integration strategy” on page 384.

E.3 WebSphere and Domino integration on the AS/400 system

The WebSphere Application Server and Lotus Domino create a winning combination for developing, deploying, and managing sophisticated enterprise applications for e-business Web sites. Domino provides a Web-enabled foundation for messaging group collaboration and workflow, and WebSphere is used to manage Internet applications that use JavaBeans, Java servlets, JavaServer Pages (JSP), and Enterprise JavaBeans (EJBs). Together, WebSphere and Domino open the door to exciting possibilities for e-business solutions that leverage the strength of both technologies.

E.3.1 Reasons for using Domino and WebSphere together

Why would you want to use Lotus Domino for AS/400 and WebSphere Application Server together since both of them have a well established and reliable application server environment? Why wouldn't you just develop your solutions on a single platform either on Domino or WebSphere?

Examining both products closely, you realize that both have distinct features, which make them outstanding in what they do. Some of the features are common to both servers, but each has its own unique qualities to make it fit in a particular space in the IBM Application Framework for e-business.

E.3.1.1 Key features of Domino

Domino as an application server is optimized for:

- Application and collaborative services
- Messaging services
- Directory services
- Security services
- Calendaring services
- Indexing and searching services
- Workflow services
- Content management services
- Access control services

E.3.1.2 Key features of WebSphere

WebSphere as an application server is optimized for:

- Transactional services
- Basic HTTP Web services (IBM HTTP Server)
- Server-side Web services (Servlets, JSPs)
- Component runtime services (JavaBeans, EJBs)
- Distributed Transaction Management
- Enhanced support for Java programming model
- Performance (Network Dispatcher, Distributed File Systems)

The key features of each product and the common features to both are illustrated in Figure 269 on page 384.

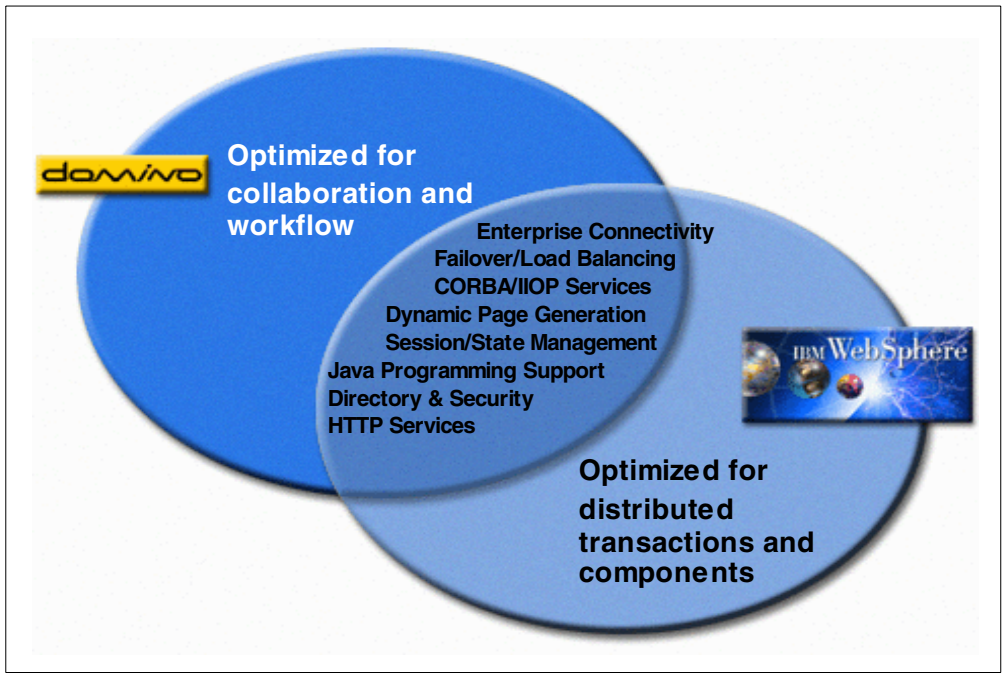


Figure 269. Features of Domino and WebSphere

Some applications fit perfectly into the space for Domino, while others fit perfectly into that of WebSphere. Unfortunately some applications span both spaces. Although you can develop those applications on Domino or WebSphere, you would get better efficiency and effectiveness by combining and using both products.

E.3.2 Integration strategy

The main idea behind integrating Domino and WebSphere is to have a unified platform that uses the combined strengths of both products. Application environments that are developed on Domino would use WebSphere for the JSP support, EJB support, session state management, and transactional and distributed components implementation. The application environments that are developed on WebSphere would use the advantages of Domino for messaging, collaboration, and workflow. This is illustrated in Figure 270.

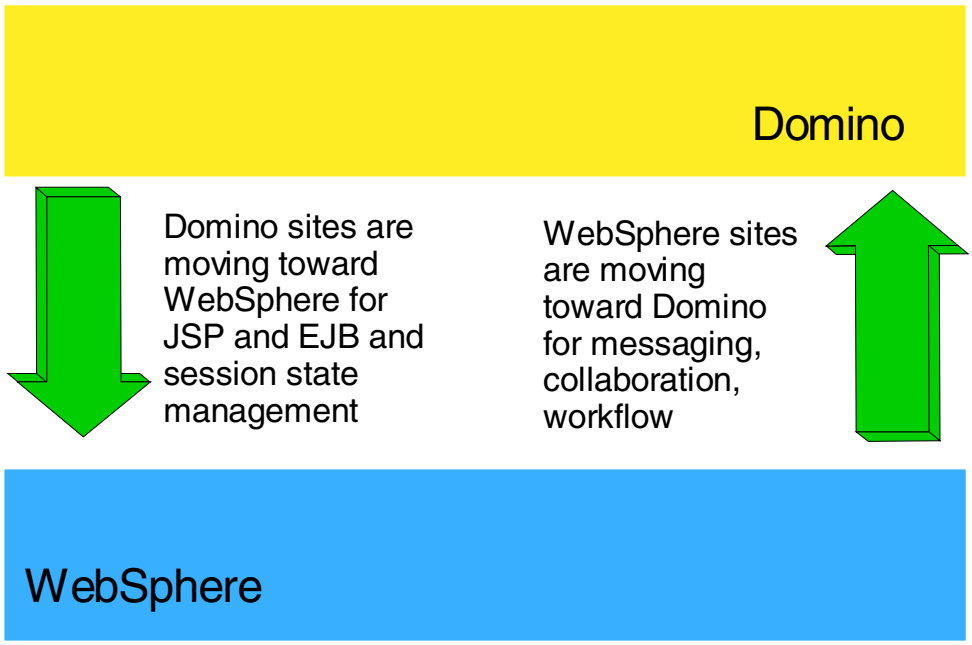


Figure 270. Trends in Domino and WebSphere integration

There are a number of ways of integrating Domino and WebSphere.

Since WebSphere has a more robust servlet engine, Domino would normally pass requests for servlets and JSPs to WebSphere. Domino agents can also access EJBs running under WebSphere by using IIOP.

Since Domino is better at handling workflow, such WebSphere objects as servlets and EJBs would also access Domino objects by using the Notes API or classes that fully represent the Domino Object Model.

This results in a scenario where Domino transfers the transactional services to WebSphere, which processes the requests using JSPs, servlets, and EJBs. When the results are ready, WebSphere either sends them to the client or sends them back to Domino for further processing.

WebSphere also uses the Notes classes to create documents and initiate workflow, collaboration, and messaging processes. After completion of the workflow processes, the results could be pushed back to WebSphere using Domino agents.

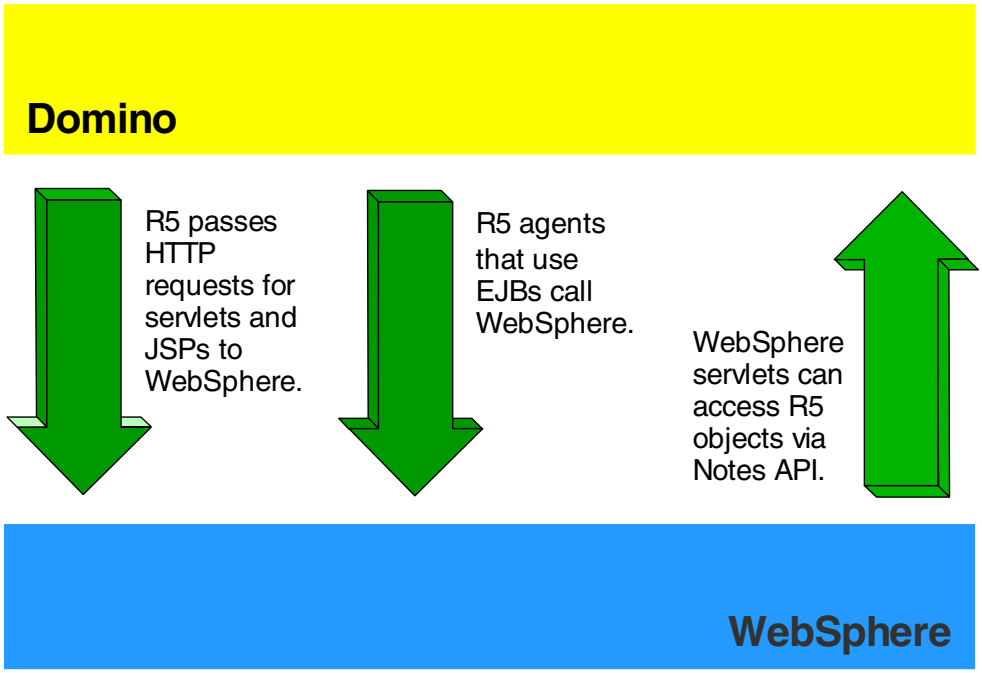


Figure 271. Interaction between Domino and WebSphere

E.3.2.1 Integration using a directory

A *directory* is a listing of information about objects arranged, in some order and gives details about each object. A common example is the yellow pages of a city telephone directory. In the computer world, a directory is a specialized database, also called a *data repository*, that stores typed and ordered information about objects. Directories allow users or applications to find resources that have the characteristics needed for a particular task.

For example, a directory of users can be used to look up a person's e-mail address, telephone number, fax number or password. A directory of application servers could be searched to find a server that can access customer billing information.

Searching a directory is similar to looking up a name in the white or yellow pages of a telephone directory. If the name of a particular individual object is not known, the directory can be searched for a list of objects that meet a certain requirement. However, directories stored on a computer are much more flexible than the yellow pages of a telephone directory because they can usually be searched by specific criteria, not just by a predefined set of categories.

Although directories may originally have been used for databases of personal information, such as a telephone number or e-mail address, the number of directory applications has recently increased considerably. Directories are now being used to hold all of the information about a person and for authenticating a user to network services or for connecting to applications.

Information in a directory is stored using the ISO standard known as X.500. X.500 organizes directory entries in a hierarchical name space capable of supporting large amounts of information.

The Lightweight Directory Access Protocol (LDAP) defines a communication protocol that defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. However, when referring to a directory that can be accessed using LDAP, the directory is usually called an LDAP directory. Therefore, LDAP directories can be implemented in many different ways. IBM implements cross-platform LDAP directories using DB2 and Lotus Domino. The implementation on DB2 is called SecureWay Directory, and the implementation in Domino is called the Domino Directory.

Note

The Domino Directory is an implementation of a Directory Server that is based on the Names and Address Book (NAB).

One way of integration between Domino and WebSphere is by using a Directory Server to identify users and groups.

There are two ways of integration using directories.

Using the Domino directory
Both Domino and WebSphere can use the Domino directory as a central point for storing user information. This can be seen in Figure 272.

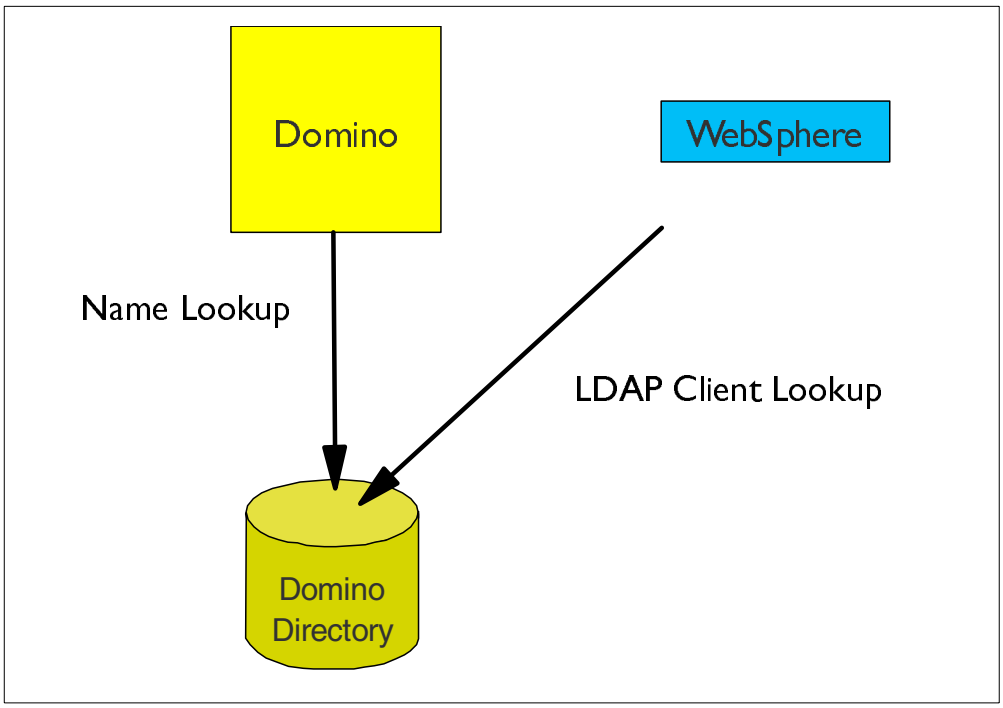


Figure 272. Using Domino as an LDAP server

To implement this type of integration, you need:

- To have the LDAP task started in Domino Server
- WebSphere Administration console to identify the LDAP Server (Domino Directory)
- Use Domino to add or change users and groups (or use existing Domino users and groups)

Using a third-party Directory server
Both Domino and WebSphere can use a third-party Directory server as a central point for storing user information. This can be seen in Figure 273.

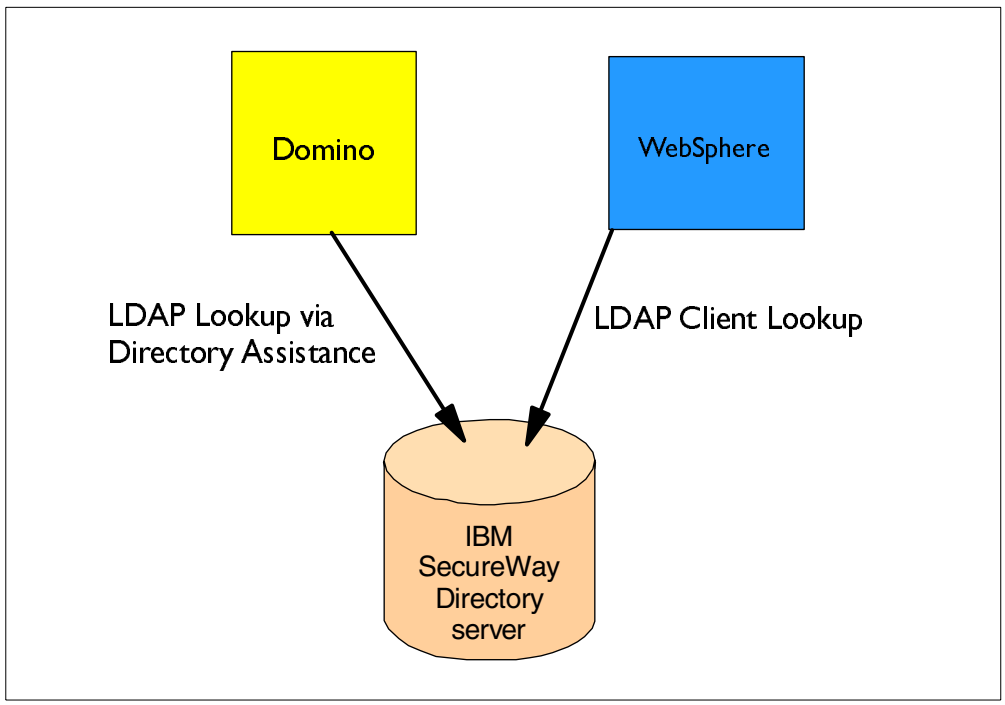


Figure 273. Using a third-party LDAP server

To implement this type of integration, you need to:

- Create a Directory Assistance database from the template file DA50.ntf.
- Set up a Directory Assistance document that tells Domino which LDAP server to search after looking in the NAB for users.
- Setup the WebSphere administration console to identify the LDAP server.

E.3.2.2 Programmatic access using Java

Due to the extensive support that both Domino and WebSphere have for Java, it is possible to access objects in both worlds using Java.

Accessing Domino objects from WebSphere
To access Domino objects from WebSphere servlets, JSPs, or EJBs, you need to have a set of Java classes that is shipped with Domino. This set of classes represent the Domino object model. If you are developing your Java application in VisualAge for Java, you have access to this set of classes, when you import the Domino access builder Library into your Workbench and then import

lotus.domino.* into your class. If you are not using VisualAge for Java, you have to get the Notes.jar file, which contains the classes. Add it to your classpath.

You access Domino objects by creating a local session object that emulates a Notes client and you use this session object to programmatically access servers, databases, views, documents, forms, and other Domino objects. To create a local session object, both Domino and WebSphere servers must be on same system.

This integration point is shown in Figure 274.

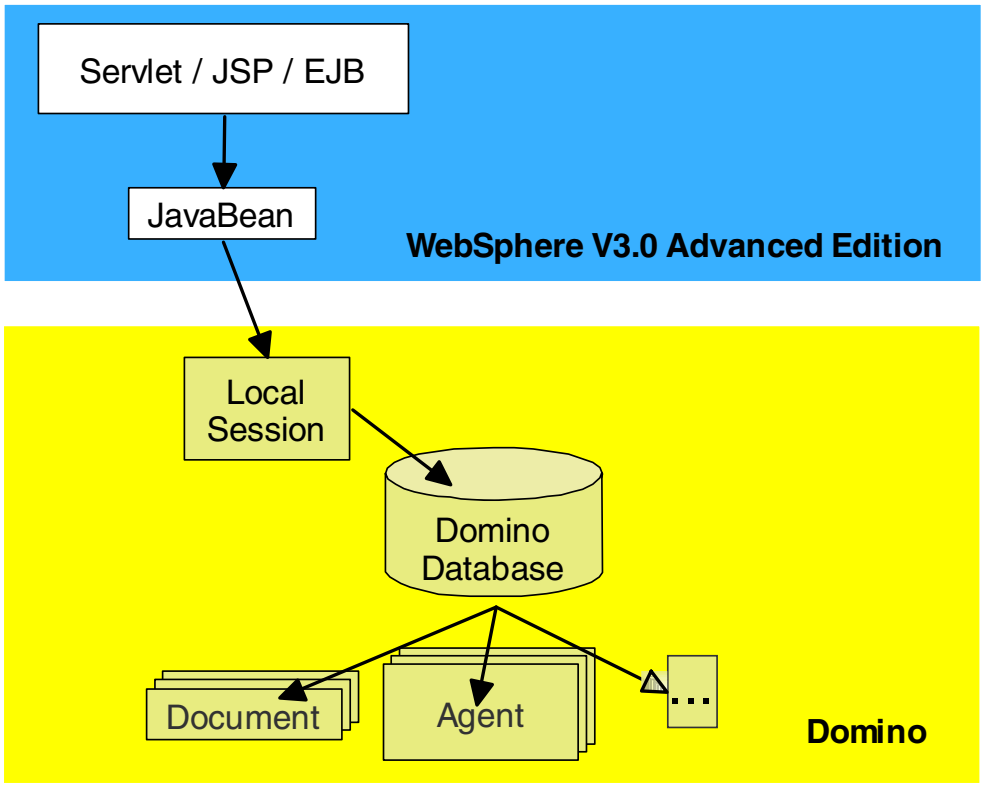


Figure 274. Accessing Domino Objects from WebSphere

Accessing WebSphere objects from Domino

To access WebSphere objects from Domino, you would normally use Remote Method Invocation (RMI) or Internet Inter-ORB Protocol (IIOP).

RMI enables a programmer to create distributed Java applications that are fully based on Java, and in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. An application that is based on Java technology can make a call on a remote object, once it obtains a reference to the remote object. It gets a reference to the remote object by either looking up the remote object in a naming service provided by RMI or by getting the reference as an argument or a return value.

IIOP is an object-oriented protocol that makes it possible for distributed programs written in different programming languages to communicate over the Internet. Using IIOP, you can develop applications that can communicate with other applications wherever they are located without having to understand anything about the program other than its service and a name.

WebSphere has services that enable it to support both RMI and IIOP. This makes it possible for Domino objects to connect WebSphere Objects.

This integration point is shown in Figure 275.

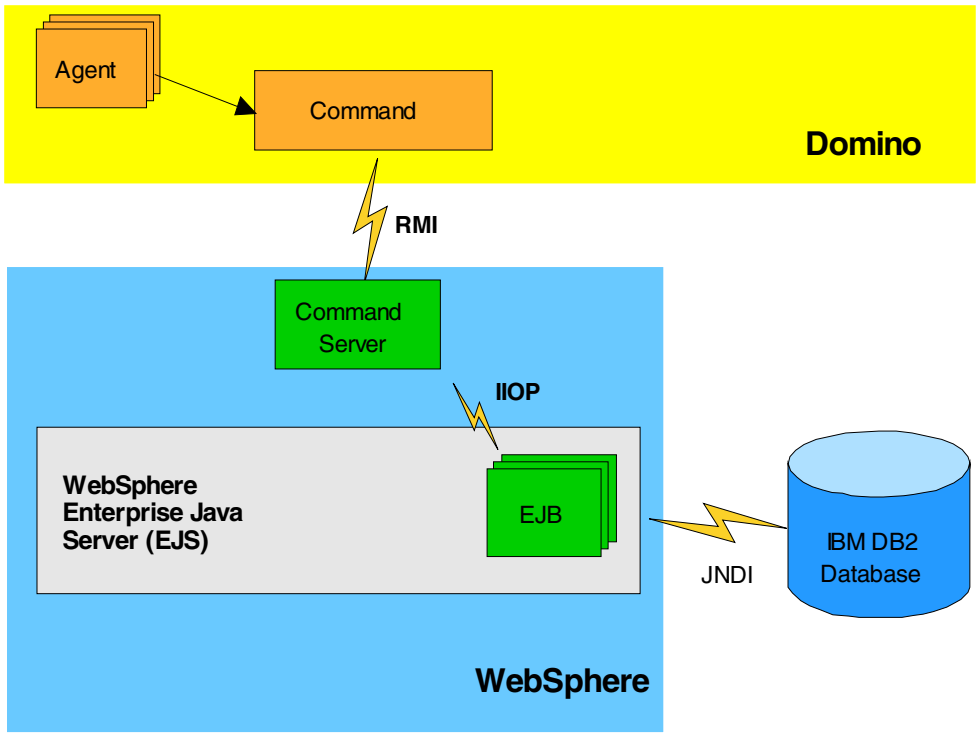


Figure 275. Accessing WebSphere objects from Domino

Development tools

There are a number of tools for developing an application that uses the integration benefits of Domino and WebSphere. Some of these tools include:

- Domino Toolkit for Java
- WebSphere Studio ships Domino ODBC and JDBC support
- Visual Age for Java ships Domino Java classes

WebSphere Studio uses the Domino for Web Content Library.

Appendix F. Domino data access control

Several levels of Domino data security are described in this section:

- Database access control
- Form access control
- Document access control
- Field access control

F.1 Database access control

Every database includes an Access Control List (ACL), which Domino uses to determine the level of access that users and servers have to that database. When a user opens a database, Domino classifies the user according to an access level that determines privileges. The access level for a user may vary in different databases.

The access level assigned to a user determines the tasks that the user can perform in the database. The access level assigned to a server determines what information the server can replicate within a particular database. Only someone with manager access can create or modify the ACL of a database located on a server.

F.1.1 Displaying the ACL

The access control list of a database shows all the servers, groups, and users who have access to the database. To display the access control list of a database, choose **File->Database->Access Control**. The display shown in Figure 276 appears.

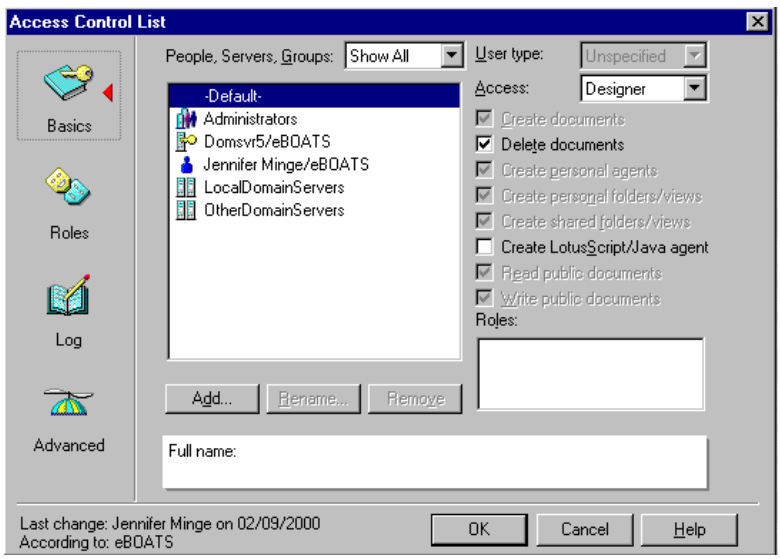


Figure 276. Database Access Control List

F.1.2 User and server access levels

A database ACL determines the level of access that users, groups, and servers have. Someone with manager access to the database assigns levels to the users, groups, and servers listed in the ACL.

With Domino Release 5.0, there are seven main levels of access that a database administrator can assign to a person, server, or group. These levels are explained in Table 36.

Table 36. The seven main level of access

Level	Users with this access can...	Servers with this access can...
No access	Not access the database at all.	Not access the replica at all.
Depositor	Create documents, but cannot read, editor delete documents, including those they create.	Not receive changes; not relevant for servers.
Reader	Read documents, but cannot create, edit, or delete them.	Pull changes from the replica but not send changes to it.
Author	Create and read documents, but can only edit their own documents if they are listed in an Authors field on that document.	Replicate a new document.
Editor	Create, read, and edit all documents unless there are restrictions on specific documents.	Replicate all new and changed documents.
Designer	Have Editor access to documents, except where restrictions exist for specific documents. They can modify the database design, but they cannot delete the database or modify the ACL.	Replicate design changes as well as all new and changed documents, but not ACL changes.
Manager	Perform all operations on the database, including modifying ACLs and deleting the database.	Replicate all changes to the database and the ACL.

F.1.3 Setting up and refining the ACL

When you set up the access control list, you can refine the access for users in several ways, beyond simply specifying an access level:

- **Select User Type to Specify Users, Groups, and Servers**: When you enter users in the ACL, you can specify whether they are users, groups, or servers.
- **Access Options**: Assigning access options allows you to further refine user access.
- **User Roles**: Roles allow you to define responsibilities in the application and refine access rights to database elements.

F.1.4 Users, groups, and servers

A group is a list of users or servers that have something in common. Using a group helps simplify many administration tasks, for example:

- A group of users can be given access to a database in the ACL.
- A group of servers can be designated as permitted to replicate with a database.
- A group of users can be denied access to a resource.

> **Note**
>
> Groups you specify in the ACL must be listed in the Domino Directory.

There are two default server groups in the ACL:

- LocalDomainServers are servers in the local domain.
- OtherDomainServers are servers in other domains. These are usually servers in other companies with whom users in your company need to communicate.

F.1.4.1 User types

The ability to specify user types lets you clearly indicate whether a name is that of a person, server, or group. See Table 37 for description of the available user types.

Table 37. Available user types

User type	Assign for this type of user	Allows you to...
Person	An individual user; this includes a user on a server workstation.	Control access for an individual user.
Server	A single server; this includes a server console, and server workstation.	Prevent someone from accessing the database from a Notes workstation using the server ID.
Server group	A group of servers.	Identify a group of servers that will host replicas of the database.
Person group	A group of individual users.	Grant the same access to all users in a group without listing each user name in the access control list.
Mixed group	A group of servers and individual users.	Grant the same access to a group of users and servers.
Unspecified	In the Advanced Access Control List window, click **Lookup User Types** for "Unspecified Users." Notes looks up an unspecified user type in the Address Book.	If you leave type as Unspecified, Domino will not check whether the access is given to a user or a server.

F.1.5 Assigning user types for additional security

Assigning user types can provide additional security. Specifying names in the ACL as a person, server, or server group prevents someone from:

- Creating a group in the Domino Directory with the same name and adding their name to it to access the database through the group name.
- Accessing the database from a Notes workstation using the server ID.

Note

Designating a name as a server or server group is not a foolproof security method. It is possible to create a Domino add-in program that gains access to the database from a workstation through the server ID, since the add-in program behaves like a server.

F.1.5.1 Access options

When you add users and groups, you can specify individual options that further refine user access. For each ACL entry, you can specify slightly different options. Table 38 describes the different access options you can use.

Table 38. Access options

Enable this option...	To allow...	This option is assigned by default to...
Create documents	Authors to create documents.	Managers, Designers, Editors, and Depositors
Delete documents	Managers, Designers, Editors, and Authors to delete documents. Authors can delete only documents they created.	No one
Create personal agents	Designers, Editors, Authors, or Readers to create personal agents.	Managers
Create private folders/views	Editors, Authors, and Readers to create personal folders and views in a database on a server.	Managers and Designers
Create shared folder/views	Editors to create shared folders and views.	Managers and Designers
Create LotusScript/Java agents	Readers, Authors, Editors, and Designers to create LotusScript and Java agents.	Managers
Read public documents*	Users to read documents created with forms, and use views and folders, designated as “available for public access user.”	Readers and above
Write public documents*	Users to create and modify documents with forms designated as “available for public access user”.	Authors and above
* Enabling users to read and write public documents lets you give users with No Access or Depositor access the ability to access specific forms, views, and documents without giving them Reader or Author access in the database. Public documents are useful for calendar applications in which one user may delegate the ability to read or create appointments on their behalf to another user.		

F.1.6 Anonymous access to databases

You can handle anonymous users in one of the following two ways:

- Define an anonymous entry in the ACL and specifically define access privileges for anonymous users.
- Allow anonymous users the same access as the Default entry in the ACL.

> **Note**
>
> Any application that will be deployed on the Web should have an *Anonymous* entry in the ACL.

If you allow anonymous access to a server, you can still control access to databases. To control database access for anonymous users, follow these steps:

1. Add a user with the name `Anonymous` in the Add User dialog box of the ACL (Figure 277).

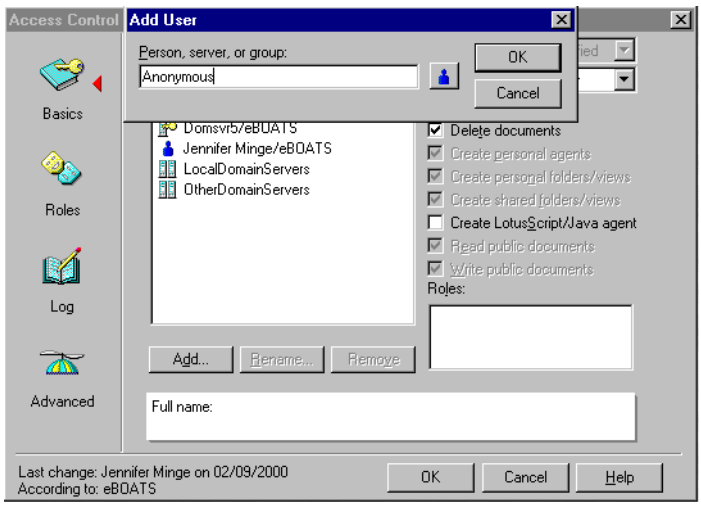


Figure 277. Adding Anonymous user to ACL

2. Click **OK**.
3. In the Access drop-down box (Figure 278 on page 396), select either:
   - **No Access** to prevent access by anonymous users
   - **Reader** to allow access to an information database
   - **Author** to allow access to an interactive database

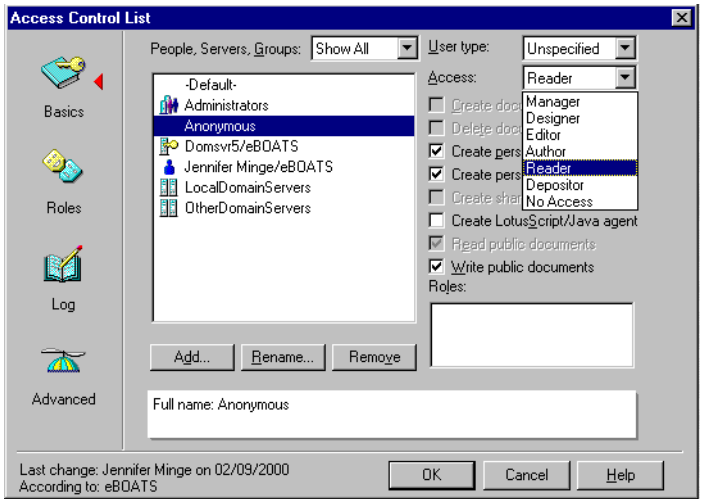


Figure 278. Access option list

Note

If the database ACL does not contain an Anonymous entry, all anonymous users receive the Default access.

To protect the databases from unregistered users, you can establish the Default as No Access. If Default access needs to be higher, create an Anonymous entry in the database ACL and grant it No Access.

When granting access to unauthenticated Web clients, you need to grant anonymous users the least access that still allows them to use the database effectively. For example, you may grant anonymous users:

- Reader access for an information database
- Author access for an interactive database

F.1.6.1 Differentiating Default and Anonymous access

If Anonymous is not listed in the ACL, Domino grants access to the user based on the default database access level. This may be a higher access level than you want for anonymous users.

The Access level definitions are:

Default A user not specified in the ACL
Anonymous A user without a valid Notes ID for that organization

F.1.7 Roles in the ACL

When a group you want to add to the ACL does not exist in the Domino directory, you may want to create a special group or role for users of the database. Roles let you define responsibilities in the application and further define access to database elements.

F.1.7.1 What a role is

A role is a subset of the ACL that is controlled by the database manager. A role can be used anywhere that a group or user name can be used. Users and groups

are assigned roles to refine access to particular views, forms, sections, or fields of a database. Instead of assigning access to a design element to users and groups, you assign access to the role.

Some advantages of using roles are that they:

- Provide a flexible method of restricting document access to a specific set of users
- Can be used in formulas
- Provide group control if you do not have the authority to create groups in the Domino Directory, or if you want to create groups just for the database
- Make it easier for you to modify access when users leave or new users join

To use a role in an application, assign roles to users and groups in the ACL. Include the role in access lists, just as you do with users and groups (or actually instead of adding specific users and groups).

F.1.7.2 Adding roles to the ACL

To add roles to an ACL, follow these steps:

1. Open the database ACL.
2. Click **Roles** in the Contents pane.
3. Click **Add**. The Add Role dialog box appears (Figure 279).

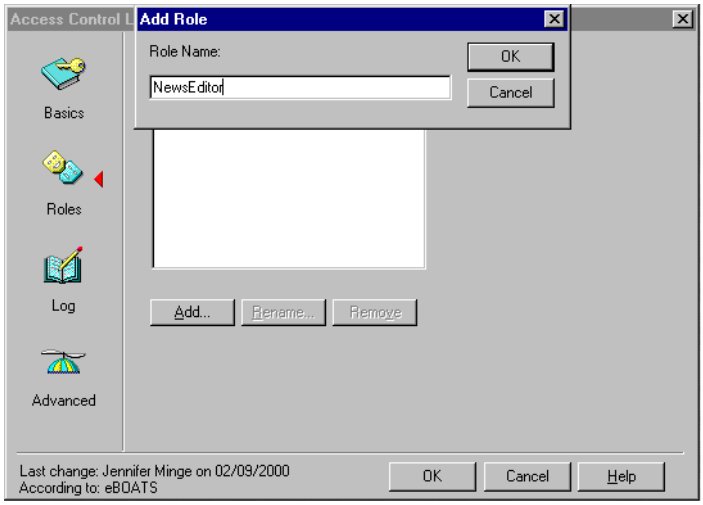


Figure 279. Add Role to the ACL

4. Enter a role name no longer than 15 characters, and click **OK**. The role name appears in brackets in the Role list.

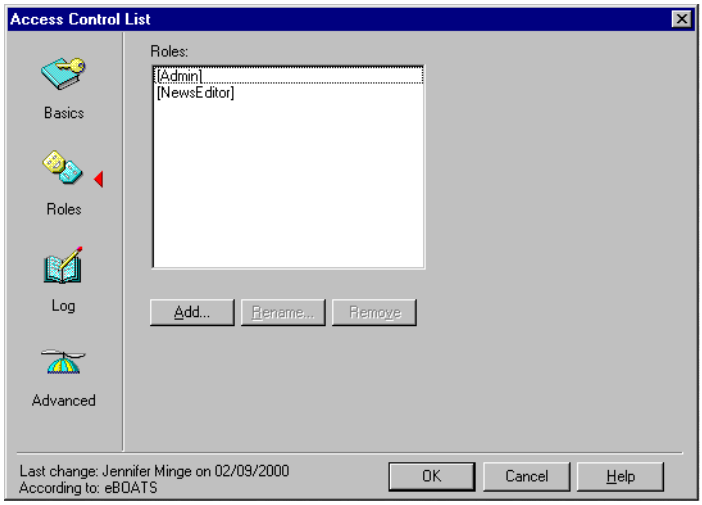


Figure 280. ACL Role list

Assigning roles to users

To assign a role to a user, complete these tasks:

1. Open the database ACL.
2. Select the user name in the list of people, servers, and groups.
3. Click one or more role names in the Roles list.

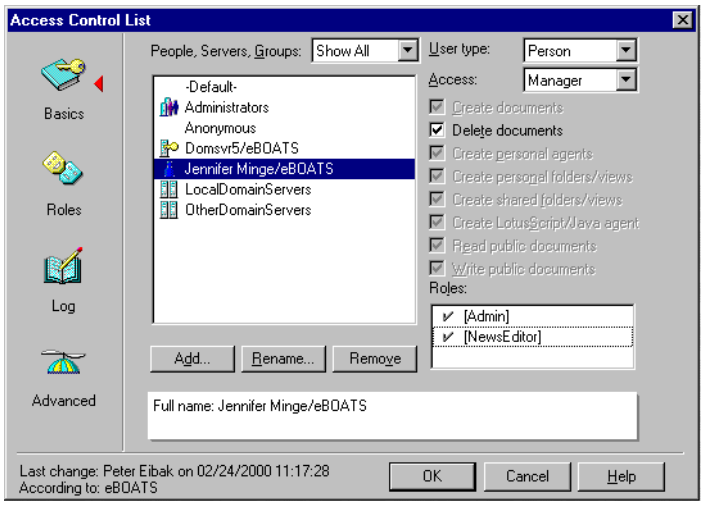


Figure 281. Assigning roles to users

4. Confirm roles by highlighting a user. A checkmark appears next to the user role or roles.

F.1.8 Enforcing consistent ACL

You can ensure that the ACL of a database remains the same on all replicas. You do this by selecting the advanced access control list option “Enforce a consistent Access Control List across all replicas of this database.” Selecting this option ensures that the ACL remains consistent across server replicas and that the ACL is enforced on replicas of the database made on workstations or laptops. If you do not select this option, users have Manager access to local replicas of server

databases, which allows them to make changes to their access levels on the server replica, although they can't replicate such changes back to the server.

You will find the option under advanced settings of the ACL as shown in Figure 282.

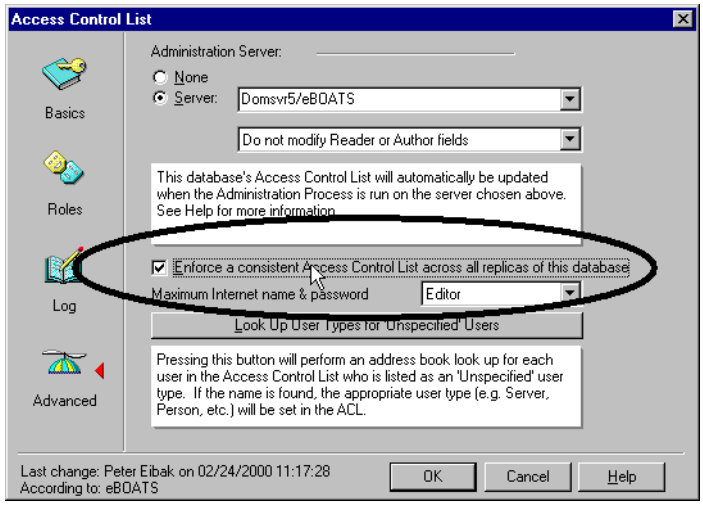


Figure 282. Enforcing a consistent ACL across all replicas of this database

Note

Enforcing a consistent access control list as it applies to ACLs on workstation or laptop replicas is not a security feature. Data in the local replica is not secure unless you physically secure the workstation or laptop or you encrypt the database using the local security feature. Also, a Domino add-in program can bypass an ACL enforced on local workstations.

To keep the ACL the same across all server replicas of a database, you must select this setting on a replica whose server has Manager access to the other replicas. Otherwise, replication will fail because the server has inadequate access to replicate the ACL.

F.1.9 Maximum Internet name and password access

When working with advanced ACL options, you can also specify a maximum access level for users that have been authenticated with the Internet name and password setting (browser users). This setting overrides individual settings in the ACL. No browser user can obtain higher access than specified for Maximum Internet Name and Password Access. Figure 283 shows where to find the setting in the ACL.

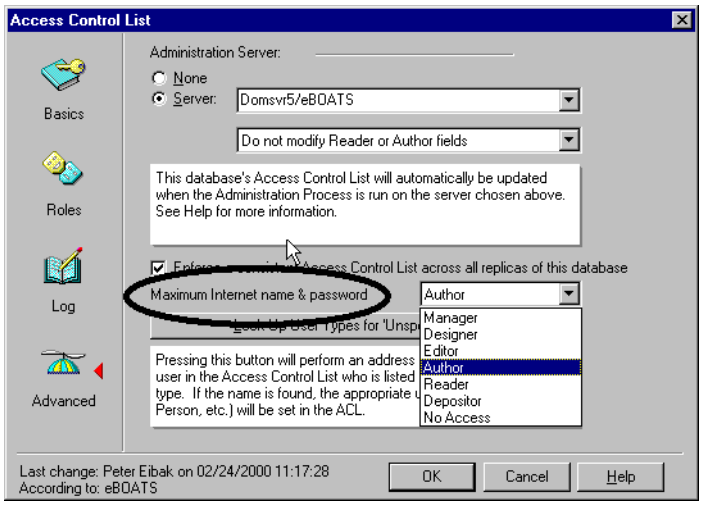


Figure 283. Maximum Internet name and password ACL setting

Tip

Check this setting if you are experiencing problems with Web users not getting the access they were granted in the ACL.

F.2 Form access control

Default read and create access to forms can be specified at the form level using the security section of the forms property box. Using the forms security refines the database ACL, allowing more flexibility in database security designs. You can restrict people who have read access to the database from reading documents created with the form. In addition, you can restrict people from creating documents with this form even though they have author (or above) access to the database. Table 39 shows which security options are available under form properties.

Table 39. Forms security options

Option	Use
Default read access for documents created with this form	Allow only a subset of users in the database ACL to read documents created with a specific form. Documents created with this form will have this subset of users as the default document reader access list.
Who can create documents with this form	Allow only a subset of users with author access or above in a database ACL to use this form to create documents.
Default encryption keys	Encryption keys are generally created by database managers and distributed to the appropriate users. Specifying this causes all encryptable fields to be encrypted when a document is saved with this form.
Disable printing, forwarding, or copying to clipboard	This feature is an aid to users to keep them from accidentally including sensitive data when reproducing a document. It is not a secure measure so it should not be relied on for true security.

F.3 Document access control

The creator of a document can determine who can read the document by using the security section of the document properties box. They can choose to allow all people with reader access or above to read the document or restrict read access to a limited number of people.

Documents inherit their read access property from the read access property in the form used to create the document. Anyone allowed to edit the document can change the document read access property.

F.3.1 Controlling access to documents using field-based access controls

Each document can have special fields defined within it that control access to it. These fields are reader fields, author fields, and signed fields.

F.3.1.1 Reader fields

Reader fields can be created on a form to allow read access restrictions to documents. A reader field consists of a list of names to be allowed read access to the document. It can be used instead of or in conjunction with form and document security read access lists. If both read access lists and reader fields are present, the users who can read the document are the addition of both lists. A reader field cannot allow access to a user that does not have read access to the database. However, it can prevent read access from someone in the database ACL that could normally read the document.

Designating a field as a reader field and as editable allows the designer to choose options for presenting name lists to the author of the document. The field could present a list of names from the address dialogs, database ACL, a view dialog, or no list so that the author could manually enter names.

A useful way to use a reader field would be to create the field so the author of a document is presented with a list of people or groups to choose from, allowing the author to determine the reader list on a document-by-document basis.

F.3.1.2 Author fields

Author fields can be created on a form to give users with author access edit capabilities to a document they didn't create. Author fields contain the names of users and are created in the same way as reader fields. This applies only to users with author access to the database.

F.3.1.3 Signed fields

Signed fields can be created on a form to allow a digital signature to be attached when a document is saved or mailed. Digital signatures verify that authors are who they say they are and guarantee that the data in the document has not been tampered with. The private key in a user ID file generates the signature. When a user opens the document containing the signed field, Notes verifies the signature by comparing it with the author's public key in the Public Address Book.

F.4 Restricting access to sections within a document

Standard sections are used in a form to collapse or expand information. They can be hidden based on whether a document is in read or edit mode or based on a

formula. This only hides a section from view. It does not protect it from updating by agents, actions, or access from another form.

Access-controlled sections are used to group areas of a form and control edit access to objects in that area. However, like the standard sections, these fields may be edited from other forms, actions, or agents.

Layout regions are similar to sections. They are areas of grouped objects that can be easily moved and displayed in ways not available with forms and subforms. As with sections, they can be hidden based on whether a document is in read or edit mode, or based on a formula. This only hides a layout region from view. It does not protect it from updating by agents, actions, or access from another form.

F.5 Field access control

Database designers can design fields that can be encrypted with an encryption key. To decrypt and read the document, users must have the same key.

Fields may also be protected during form design from updating by authors after the initial document is created. Field property security options include an option specifying that a user must have at least editor access to use the field.

Appendix G. Using the additional material

The Domino databases, Java code, and the AS/400 library used in the examples in this redbook are available to be downloaded through the Internet. The Domino examples were developed using Domino Designer 5.0.2b, VisualAge for Java V3.0, and WebSphere Studio V3.0. This appendix contains instructions on using or downloading the material.

G.1 Locating the additional material on the Internet

The Web material associated with this redbook is also available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG246052`

Alternatively, you can go to the IBM Redbooks Web site at:

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

Select the **Additional materials** link and open the directory that corresponds with the redbook form number.

G.2 Using the Web material

The additional Web material that accompanies this redbook includes:

File name	*Description*
readme.txt	Instructions for restoring the AS/400 library and the Domino databases and Java objects
sg246052.zip	Zipped files

G.2.1 System requirements for downloading the Web material

The following system configuration is recommended for downloading the additional Web material or running the examples:

Operating System: OS/400 V4R4 or later
Software: Domino for AS/400 5.0.2a or later, SQL/400

G.2.2 How to use the Web material

Create a subdirectory (folder) on your workstation and unzip the contents of the Web material into this folder.

Appendix H. Special notices

This publication is intended to help information systems architects and Lotus Notes developers understand how to conceive and develop an e-business application in a Domino for AS/400 environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM AS/400 software, including OS/400, or Lotus Domino, or Lotus Enterprise Integrator. See the PUBLICATIONS section of the IBM Programming Announcement for AS/400 software, including OS/400, for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

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

AS/400	AS/400e
AT	CICS
CT	Current

Appendix I. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

I.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 411.

- *Lotus Domino for AS/400: Performance, Tuning, and Capacity Planning*, SG24-5162
- *Lotus Domino for AS/400: Installation, Customisation, Administration*, SG24-5181
- *V4 TCP/IP for AS/400: More Cool Things than Ever*, SG24-5190
- *The AS/400 NetServer Advantage*, SG24-5196
- *Lotus Domino Release 5.0: A Developer's Handbook*, SG24-5331
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, SG24-5341
- *Lotus Domino for AS/400: Integration with Enterprise Applications*, SG24-5345
- *AS/400 Internet Security: Implementing AS/400 Virtual Private Networks*, SG24-5404
- *Slicing the AS/400 with Logical Partitioning*, SG24-5439
- *Lotus Domino for AS/400 R5: Implementation*, SG24-5592
- *Lotus Domino 5.0 Enterprise Integration: Architecture and Products*, SG24-5593
- *Performance Considerations for Domino Applications*, SG24-5602
- *Building AS/400 Applications for WebSphere 2.0 Standard Edition*, SG24-5635
- *AS/400 Mail: Multiple SMTP Domain Names Behind a Firewall*, SG24-5643
- *AS/400 Internet Security: Developing a Digital Certificate Infrastructure*, SG24-5659
- *AS/400e e-business Handbook*, SG24-5694
- *Lotus Fax for Domino for AS/400: Getting the Straight Facts*, SG24-5941
- *Lotus Domino for AS/400: Problem Determination Guide*, SG24-6051
- *Building AS/400 Applications for WebSphere Advanced 3.0*, SG24-5691

  This publication is only available online in softcopy format at the redbooks home page at: `http://www.redbooks.ibm.com`

  At the site, click **Redbooks Online!** Then, enter the title or order number in the Redbook Search field and click **Submit Search**. When the search results appear, click the appropriate title.

I.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

I.3 Other resources

These publications are also relevant as further information sources:

- *HTTP Server for AS/400 Webmaster's Guide*, GC41-5434
- *Tips and Tools for Securing Your AS/400*, SC41-5300
- *Security - Basic V4R1*, SC41-5301
- *OS/400 Security - Reference*, SC41-5302
- *AS/400 Work Management V4R4*, SC41-5306
- *DB2 UDB for AS/400 SQL Programming*, SC41-5611
- *Integrated File System Introduction*, SC41-5711
- *AS/400 Performance Capabilities Reference*, SC41-0607

  This publication is only available online softcopy format at the AS/400 Online Library at: http://as400bks.rochester.ibm.com/pubs/html/as400/onlinelib.htm

  At the site, select your language and click **GO!** Click **V4R4** and then click **Search or view all V4R4 Books**. In the search field that appears, enter the title or publication number and click **Find**. When the search results appear, click the appropriate title.

- *The View, Technical Journal for Lotus Notes and Domino* (articles). WIS, Inc.:
  - *Unlocking the Secrets of Internet Security: How to Implement X509 Certificates for SSL and S/MIME in Domino* (November/December 1999)
  - *18 Techniques for Building a Domino-Powered Web Site* (September/October 1999)
  - *Performance-Engineering Domino Application Designs for Notes and Web Clients* (September/October 1999)
  - *How to provide Browser Users with Personalized Views on an Intranet Web Site* (July/August 1999)
  - *Java for Domino: Why?* (July/August 1999)
  - *Programming for Performance with LotusScript and Java* (May/June 1999)

  - – *Hello World! Getting started with JavaScript in Domino* (May/June 1999)
  - – *Programming for Performance with LotusScript and Java* (May/June 1999)
  - – *Programming Remote Java Applications and Applets for Domino R5* (November/December 1998)
  - – *Introducing R5 Support for JavaScript and the New Java Agent Editor* (November/December 1998)
  - – *Building Java Applications and Agents with Domino 4.6.* (November/December 1997)
- Lotus White Paper *Maximizing Application and Server Performance in Domino (PDF)* available at:
  `http://www.as400.ibm.com/developer/domino/perform/maxperform.pdf`
- The White Paper *AS/400: 99+% Availability*, can be accessed at:
  `http://www.as400.ibm.com/whpapr/999.htm`
- Mason, Jim. “Using Domino for Web applications”, *AS/400 Experts Journal*. Vol. 2, No. 3, September 1999.
- *Platform Availability Data: Can You Spare a Minute?*, a document published by the Gartner Group, can be viewed online at:
  `http://www.gartner.com/webletter/ibmglobal/edition2/article4/article4.html`
- *Lotus Domino Designer Release 5*, Lotus Part Number CT6E2NA

  This book is available from Lotus. You can contact Lotus on the Web at:
  `http://www.lotus.com`

I.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- Visit the Lotus Developer Central Web site at:
  `http://www.lotus.com/home.nsf/welcome/developernetwork`
- AS/400 Consultant reports, a TCO analysis of IBM AS/400 Dedicated Server for Domino versus PC Servers resource, can be accessed online at:
  `http://www.as400.ibm.com/conslt/dsd_tco.htm`
- For Notes application design performance information, see the Web site at:
  `http://www.notes.net`
- The white paper “Evaluating Appropriate Workloads for the AS/400e”, which explores the behavior and performance of light data integration on the same server to help with your planning and analysis, can be found at:
  `http://www.as400.ibm.com/whpapr/dsd.htm`
- For sizing information geared toward basic uses of Domino, such as e-mail and out-of-the-box template applications, such as discussion databases, refer to the document *Sizing Domino for AS/400*, which is available online at:
  `http://www.as400.ibm.com/domino/domsz2.htm`
- The IBM AS/400 Authorized Problem Analysis Reports database can be accessed at: `http://as400service.ibm.com`

  On the left pane, click **Tech Info & Databases** and then **Software Problems - APARS**. A full text search is provided.
- Visit the Notes home page at: `http://www.notes.net/welcome.nsf?OpenDatabase`

- For information on adjusting the performance on your system, see the article *Performance Tuning of AS/400 for Highly Threaded Applications*. You can find this article at:
  http://www.as400.ibm.com/tstudio/tech_ref/perftune/Threads.htm
- Visit the following Web site to search through the hundreds of Domino applications that are ported to the AS/400 system:
  http://www.as400.ibm.com/developer/domino
- IBM Workload Estimator for AS/400, a Java-based tool for sizing an AS/400 server for a mixed workload, including Domino, can be accessed online at:
  http://as400service.ibm.com/estimator
- Sizing information for Lotus Domino for AS/400 is available online at:
  http://www.as400.ibm.com/domino/D4szintro.htm
- For performance hints and tips on Web forms, views, and application design, refer to the Lotus white paper *Maximizing Application and Server Performance in Domino* (in PDF format) at:
  http://www.as400.ibm.com/developer/domino/perform/maxperform.pdf
- For sizing information about more advanced Domino applications, refer to the document *Domino for AS/400: Application Sizing Examples* online at:
  http://www.as400.ibm.com/domino/D4appsz.htm
- The Domino for AS/400 Release Notes (readas4.nsf) database contains last-minute information specific to the AS/400 platform. A printed version comes with the Domino for AS/400 CD-ROM. The following Web site may have an updated version: http://notes.net/doc
- For information about the SSL protocol and the SSL handshake, refer to the SSL 3.0 Internet draft at: http://www.netscape.com/eng/ssl3/index.html
- There is a special MIME format, application/x-x509-ca-cert, which allows a browser to receive a new CA certificate that has been signed by one of the known CAs. This format is specified in PKCS #7 and is available at:
  http://www.rsa.com/rsalabs/pubs/PKCS/index.html
- The Enterprise Integration Web page is a rich source for information, downloads, white papers, future downloads, discussion databases, and patches. It can be accessed at: http://www.eicentral.lotus.com
- The Domino Design Components can be downloaded free of charge at:
  http://www.lotus.com/Webauthor
- For customers who want to use the euro currency symbol on Windows, a free update is available from Microsoft from the following Web address:
  http://microsoft.com/windows/euro.asp

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

@DB function A fast and easy-to-use read-only access method to ODBC-compliant DBMSs.

access control list (ACL) (1) A collection of all access rights for one object. (2) A list associated with an object that identifies all subjects that can access the object and their access rights, for example, a list associated with a file that identifies users who can access the file and their access rights to that file.

application programming interface (API) A set of calling conventions defining how a service is invoked through a software package.

agent A Lotus Notes routine that automates tasks. Agents generally perform routine Lotus Notes tasks in the background but can also be started interactively by a user. You can program agents, using Java, LotusScript, or the Lotus Notes formula language.

application unit of work A set of actions within an application that the designer chooses to regard as an entity. It is up to the designer to decide how, if at all, an application should be subdivided into application units of work, and whether any application unit of work will consist of one, or many, logical units of work (LUWs).

authentication (1) Verification of the identity of a user or the user's eligibility to access an object. (2) Verification that a message has not been altered or corrupted. (3) A process used to verify the user of an information system or protected resources.

authorization (1) The right granted to a user to communicate with or use a computer system. (2) An access right. (3) The process of granting a user either complete or restricted access to an object, resource, or function.

certificate A message signed with a public key digital signature stating that a specified public key belongs to someone or something with a specified name.

certification authority An entity, typically a company, that issues digital certificates to other entities (organizations or individuals) to allow them to prove their identity to others

certification In Lotus Notes, the process of having an authorized (the certifier) person authenticate the identity of a user or server.

call level interface (CLI) A callable API for database access, which is an alternative to the embedded SQL API. In contrast to embedded SQL, CLI does not require precompiling or binding by the user but instead provides a standard set of functions to process SQL statements and related services at runtime.

client As in client/server computing, the application that makes requests to the server and often deals with the interaction necessary with the user.

client/server computing A form of distributed processing, in which the task required to be processed is accomplished by a client portion that requests services and a server portion that fulfills those requests. The client and server remain transparent to each other in terms of location and platform. See *client*, *distributed processing*, and *server*.

commit An operation that applies all changes made during the current unit of work.

database (1) A collection of interrelated data stored together with controlled redundancy according to a scheme to serve one or more applications. (2) All data files stored in the system. (3) A set of data stored together and managed by a database management system. (4) In Lotus Notes, a group of documents and their forms and views, stored under one file.

distributed processing An application or systems model in which function and data can be distributed across multiple computing resources connected on a local area network or wide area network. See *client/server computing*.

Distributed Relational Database Architecture (DRDA) Architecture proposed by IBM, which defines the rules, the protocols and the semantics for writing programs implementing distributed data access. All the platforms participating in this architecture must comply with these rules and definitions.

document In Lotus Notes, an object containing text, graphics, video, or audio data or any kind of rich text data.

Domino Lotus Domino is an applications and messaging server with an integrated set of services that enable you to easily create secure, interactive business solutions for the Internet and corporate intranets.

electronic messaging The creation, transfer, storage, and retrieval of text, graphics, image, or voice data by electronic means.

environment The collective hardware and software configuration of a system.

Extensible Markup Language (XML) A standard metalanguage for defining markup languages that was derived from and is a subset of SGML. XML omits the more complex and less-used parts of SGML and makes it much easier to (a) write applications to handle document types, (b) author and manage structured information, and (c) transmit and share structured information across diverse computing systems. The use of XML does not require the robust applications and processing that is necessary for SGML. XML is being developed under the auspices of the World Wide Web Consortium (W3C).

form In Lotus Notes, forms are used to author, read, and edit documents. Application developers design forms as templates for document creation, or as a way to view document data. Forms are comprised of other design elements such as text, fields, buttons, and programming code.

formula A Lotus Notes programming language that contains a set of built-in macros, functions, and commands.

graphical user interface (GUI) A style of user interface that replaces the character-based screen with an all-points-addressable, high-resolution graphics screen. Windows display multiple applications at the same time and allow user input by means of a keyboard or a pointing device such as a mouse, pen, or trackball.

Hypertext Markup Language (HTML) A markup language that conforms to the SGML standard and was designed primarily to support the online display of textual and graphical information that includes hypertext links.

Hypertext Transfer Protocol (HTTP) In the Internet suite of protocols, the protocol that is used to transfer and display hypertext documents.

host (1) In a computer network, a computer providing services such as computation, database access, and network control functions. (2) The primary or controlling computer in a multiple computer installation.

IIOP Internet Inter-ORB Protocol.

index An ordered set of pointers to the data of a DBMS table. An index allows more efficient access to rows in a table by creating a direct path to the data through the pointers. Each index is stored separately from the table it refers to and is based on the values of data in one or more columns of the table.

intranet A TCP/IP network that is entirely under the control of a private authority or company. The intranet may have connections to other independent intranets (which would then be referred to as *extranets*) or the Internet. It may be fully or partially visible to the outside, depending on the implementation.

Internet Large international, national, and regional backbone networks that allow local and campus networks and individuals access to global resources.

JavaBeans A platform-independent, software component technology for building reusable Java components called “beans”. Once built, these beans can be made available for use by other software engineers or can be used in Java applications. Also, using JavaBeans, software engineers can manipulate and assemble beans in a graphical drag-and-drop development environment.

Lightweight Directory Access Protocol (LDAP) An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). Applications that use LDAP (known as directory-enabled applications) can use the directory as a common data store and for retrieving information about people or services, such as e-mail addresses, public keys, or service-specific configuration parameters.

link A defined connection from the NotesPump server to a specific database product with specific access parameters, such as user IDs and passwords.

LotusScript The Lotus cross-product BASIC scripting language.

logical unit of work (LUW) (1) An update that durably transforms a resource from one consistent state to another consistent state. (2) A sequence of processing actions (for example, database changes) that must be completed before any of the individual actions can be regarded as committed. When changes are committed (by successful completion of the LUW and recording of the syncpoint on the system log), they do not have to be backed out after a subsequent error within the task or region.

LotusScript extension (LSX) A set of specialized LotusScript classes that extends the standard object model to meet a specific need (for example: access to a DBMS or to an ERP).

metadata A defining unit of data such as a database table, data file, or Notes form.

metalink A special kind of NotesPump link that provides preprocessing operations on link data before it is transferred within a defined Activity form. Two metalinks, Order and Collapse/Expand, are supplied with NotesPump 2.5.

middleware A set of services that allow distributed applications to interoperate on a local area network or wide area network. It shields the developer or end user from the system complexity and enables the delivery of service requests or responses transparently across computing resources.

MIME See *Multipurpose Internet Mail Extensions*.

Multipurpose Internet Mail Extensions (MIME) An Internet standard for identifying the type of object being transferred across the Internet. MIME types include several variants of audio, graphics, and video.

Open Database Connectivity (ODBC) A Microsoft developed C database API that allows access to DBMSs through callable SQL, which does not require the use of an SQL preprocessor.

Public-Key Cryptography Standards (PKCS) A series of documents produced and distributed by RSA Security, Inc., proposing techniques for using public key cryptographic algorithms in a safe and interoperable manner.

recovery The use of archived copies to reconstruct files, databases, or complete disk images after they are lost or destroyed.

replication A Lotus Notes procedure that updates and distributes copies (replicas) of the same Lotus Notes database that are stored on different servers.

rich text A Lotus Notes field capable of storing a variety of type styles, graphics, and multimedia.

Secure Sockets Layer (SSL) A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

server Any computing resource dedicated to responding to client requests. Servers can be linked to clients through local area networks or wide area networks to perform services, such as printing, database access, fax, and image processing, on behalf of multiple clients at the same time.

Simple Mail Transfer Protocol (SMTP) In the Internet suite of protocols, an application protocol for transferring mail among users in the Internet environment. SMTP specifies the mail exchange sequences and message format. It assumes that the Transmission Control Protocol (TCP) is the underlying protocol.

spoofing Convincing someone that you are entity X when you are not X, without X's permission. Synonyms are impersonate and masquerade.

Standard Generalized Markup Language (SGML) A standard metalanguage for defining markup languages that is based on the ISO 8879 standard. SGML focuses on structuring information rather than presenting information. It separates the structure and content from the presentation. It also facilitates the interchange of documents across an electronic medium.

structured query language (SQL) A standard set of statements used to manage information stored in a database. By using these statements, users can add, delete, or update information in a table, request information through a query, and display the result in a report.

stored procedures A facility to execute procedures that are stored at the server. Stored procedures allow an application program to be run in two parts. One part runs on the client, and the other part runs on the server. Therefore, one call to a remote database can execute a procedure that may represent several repetitive accesses to the database. The server procedure at the database runs within the same transaction as the client application.

table A relational database presents data as a collection of tables. A table consists of data logically arranged in columns and rows. The data in the table is logically related, and relationships can be defined between tables. Data can be viewed and manipulated on the basis of mathematical principles and operations called *relations*. Table data is accessed through SQL.

Transmission Control Protocol/Internet Protocol (TCP/IP) A set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

transaction A unit of processing (consisting of one or more application programs) initiated by a single request. A transaction can require the initiation of one or more tasks for its execution.

transaction processing A style of computing that supports interactive applications in which requests submitted by users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A transaction processing system supervises the sharing of resources for processing multiple transactions at the same time.

Trojan horse A piece of code embedded in a useful program for devious purposes, for example, to steal information. Usually the term Trojan horse is used rather than virus when the offending code does not attempt to replicate itself into other programs.

trusted server A server that aids in network authentication.

unit of work A recoverable sequence of operations performed by an application between two points of consistency.

Uniform Resource Locator (URL) The Internet address for a document, file, or other resource. It describes the protocol required to access the resource, the host where it can be found, and a path to the resource on that host.

view (1) In Lotus Notes, views are used to sort, list, filter, and categorize documents. Application developers design views with particular selection criteria so that they will display lists of pertinent documents in a given order. (2) In a relational DBMS, a view is an efficient way of representing data without having to maintain it. It can include all or some of the columns or rows of the tables on which it is defined.

Web See *World Wide Web*.

Web browser The client component of the World Wide Web. The Web browser is responsible for formatting and displaying information, interacting with the user, and invoking external viewers for data types that it does not support directly. Examples of Web browsers are Netscape Communicator and Microsoft Internet Explorer.

Web page Any document that can be accessed by a Uniform Resource Locator (URL) on the World Wide Web.

Web server The server component of the World Wide Web. It is responsible for servicing requests for information from Web browsers. The information can be a file retrieved from the server's local disk or generated by a program called by the server to perform a specific application function.

World Wide Web (WWW) (W3) (the Web) An Internet client/server distributed information and retrieval system based on HTTP that transfers hypertext documents across a varied array of computer systems.

workstation (1) A configuration of input and output equipment at which an operator works. (2) A terminal, a microcomputer or a network computer, usually one that is connected to a server, a mainframe or a network, at which a user can perform applications.

X.400 A CCITT standard for electronic mail.

X.500 The directory services standard of ITU, ISO, and IEC.

X.509 A CCITT standard for security services within the X.500 directory services framework. The X.509 encoding of public key certificates has been widely adopted. The other protocol elements of X.509 have not been adopted.

XML See *Extensible Markup Language*.

Index

Symbols

A

B

C

N

O

P

Q

R

S

T

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6052-00 Developing e-business Applications: Using Lotus Domino for AS/400:
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

Developing e-business Applications Using Lotus Domino for AS/400

(0.5” spine)
0.5”<->0.875”
250 <-> 459 pages

Developing e-business Applications

Using Lotus Domino for AS/400

Implement Domino for AS/400 in your e-business strategy

Develop, setup, and secure your e-business application

Discover practical new examples

Today, many AS/400 and non-AS/400 customers are given the project of developing an e-business application, which exploits the latest technology standards. Using Lotus Domino for AS/400 is one of the solutions that you'll want to consider.

This redbook is designed to help information systems architects and Lotus Notes developers conceive and develop an e-business application in a Domino for AS/400 environment. It defines for you the valuable role that Domino for AS/400 plays in such a project and explains the tools and processes that are involved in setting it up.

Plus, it offers you a practical understanding, through an example, of how to create an e-business application running on Domino for AS/400. The sample application demonstrates functions open to public users (Internet), functions available only to authorized business partners (extranet), and functions available only to employees (intranet). The application is used mainly through a Web browser and is protected using a large panel of security features, including SSL.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by IBM's International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6052-00 ISBN 0738417807