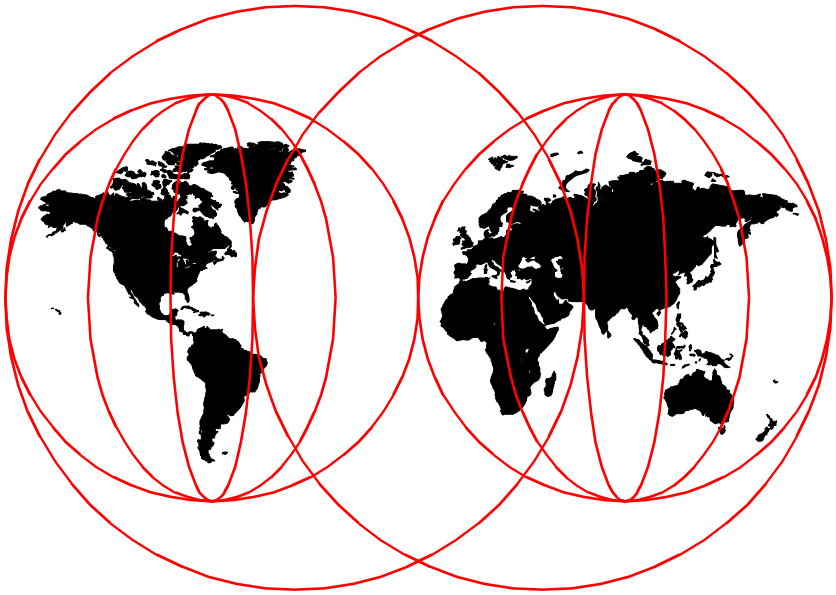
Lotus

IBM

# From Client/Server to Network Computing
# A Migration to Domino

*Christophe Toulemonde, Toyoki Matsumara, Andreas Reichert*



**International Technical Support Organization**

http://www.redbooks.ibm.com

SG24-5087-00

SG24-5087-00

International Technical Support Organization

# From Client/Server to Network Computing
# A Migration to Domino

August 1998

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix E, “Special Notices” on page 137.

**First Edition (August 1998)**

This edition applies to Lotus Domino Version 4.6, MQSeries & CICS Connections for Domino 1.0, and Lotus NotesPump Release 2.5

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE  Building 80-E2
650 Harry Road
San Jose, California 95120-6099

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

# Contents



# Figures

# Preface

In 1992 at the international technical support organization (ITSO), San Jose Center, an international team designed and implemented a client/server application to demonstrate how the client/server computing model can be implemented with IBM's mainline transaction and database products in the OS/2 and MVS/ESA environments. At that time, the application used the best architecture, development tools and techniques, and products.

Five years later, new technologies, especially the Internet, have strongly modified the computer industry and company environments. By combining Web technologies with the vast resources of traditional information technology, companies around the world are shifting their business activities to the Internet. Doing business on the Internet is known as *e-business*.

To gain the advantage of this new network computing model and environment, companies must choose to create completely new applications to exploit the Internet or migrate and enhance existing applications to support Web technologies. These applications, allowing the creation of expandable e-business solutions, are developed using the Network Computing Framework, an open, standards-based framework. As part of this framework, Domino and the Lotus connectors provide foundation services such as mail and collaboration to e-business applications.

In this book, we explore the migration of a client/server application to this new framework, using the Domino environment. In 1997 at the ITSO - San Jose Center, an international team migrated the 1992 application to the new architecture with three major objectives:

- Reuse the maximum of the existing application
- Open the application to the Internet
- Support mobile users

We explain the complete migration process, from design to production. This book will help you understand the steps required in a migration process and the questions to ask. It describes the different tools and technologies that we used to accomplish the migration and presents some alternative tools and techniques to investigate.

This book is intended for technical professionals who are planning to use Domino to migrate a client/server application to a network computing application.

## 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 San Jose Center.

**Christophe Toulemonde** is a Senior ITSO Specialist for client/server and network computing at the Application Development and Data Management ITSO, San Jose Center. He writes extensively and teaches IBM classes worldwide on all areas of client/server and network computing. Before joining the ITSO, Christophe worked as a Technical Manager in an IBM subsidiary, Datablue, in France. You can reach him by e-mail at toulemon@us.ibm.com.

**Toyoki Matsumara** is a Lotus Notes Application Development Consultant in IBM Japan, Systems Engineering. He has four years of experience in Lotus Notes application development and deployment. Toyoki and his team are responsible for providing solutions to systems engineers and customers of IBM Japan in all aspects of Lotus Notes. You can reach him by e-mail at tma@jp.ibm.com.

**Andreas Reichert** is a Lotus Notes Application Developer in IBM Global Services, Germany. He has 6 years of experience in the field of client/server development. He has worked at IBM for 14 years. His areas of expertise include the life cycle of application development as well as the design, implementation, and administration of relational databases. Besides his engagement in IBM customer projects, Andreas is also involved with an EMEA Chief Information Officer's architecture mission that investigates the migration of host-based applications to network computing applications. You can reach him by e-mail at areicher@de.ibm.com.

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

**Fiona Collins**
IBM International Technical Support Organization, Cambridge Center

**Justine Grose**
IBM EMEA Transaction Systems CoC

**Martha Hoyt**
Lotus Product Manager - MQSeries and CICS Connections for Domino

**Brian Macfaden**
IBM International Technical Support Organization, Poughkeepsie Center

**Lauren Wendel**
Lotus Product Manager - NotesPump

Thanks also to **Maggie Cutler** for her editorial review.

## 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 “ITSO Redbook Evaluation” on page 167 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@us.ibm.com`

# Chapter 1. Foreign Currency and Traveler's Check Application

Back in 1992, we developed the foreign currency and traveler's check application, called CS92 for Client/Server, using the basic principles of the client/server model. We chose that application for the following reasons:

- The simplicity of its business concepts, as most people have bought or sold foreign currency before going on foreign travel
- The generic components of its processes, which can easily be applied to any business that involves order and supply processing
- The applicability of the business to a client/server computing architecture

In this chapter, we explain CS92 by briefly describing its functions so that you can understand the migration process we describe in the subsequent chapters.

## 1.1 Business Concepts

We introduce CS92 by describing its operation in a fictitious bank that provides foreign currency and traveler's checks to customers. CS92 is used in the bank branches by the bank's clerks and in the central department at the bank head office.

Bank branches supply currency and checks on demand, where demand justifies maintaining stocks at the branch. Otherwise, currency and checks are ordered from the central department and mailed to the customer or to the branch for collection. The branches also cash in traveler's checks and purchase currency.

The central department is responsible for maintaining appropriate stock levels centrally and at all branches. Maintaining these levels is very important for currency, because stock holdings cannot accrue interest, and insufficient stocks mean lost sales.

CS92 also deals with reconciling branch stocks against sales and purchases, exchanges rates, and calculation of branch and central department commissions.

## 1.2 Business Processes

In functional terms, CS92 covers all aspects of a bank's foreign currency and traveler's checks operation. The business processes involved are:

- Customer order management
- Cashier management
- Branch management
- Central bank management

We modeled all of the business processes but implemented only two subprocesses:

- Customer purchases from branch cashier

  This is the most frequently performed function in the application. Purchases are normally for the currency and traveler's checks of the destination country. Payment for this service can be by cash, credit card, local check, or a debit to the customer's account.

  Stock levels are reduced, a customer tab is printed, and accounting entries are passed to the accounts application.

- Branch stock replenishment

  At the end of each day, the requests of the cashiers are consolidated. Each request can be an order either to replenish stock or send excess stock. A consolidated branch order is then sent to the central departments.

## 1.3 CS92 Infrastructure

On the basis of the bank's organization, we implemented a three-level system structure:

- The client workstation, used by the cashier, running OS/2 2.0 with IBM Extended Services with Database Server for OS/2 Version 1.0 and CICS OS/2 Version 1.2
- The local server, located in the branch office, running OS/2 2.0 with IBM Extended Services with Database Server for OS/2 Version 1.0, DDCS/2 1.0 and CICS OS/2 Version 1.2
- The mainframe server, located in the head office, running MVS/ESA Version 4.2 with CICS/ESA Version 3.3 and DB2 Version 2.3

CS92 ran on a LAN-attached OS/2 workstation linked to an MVS/ESA mainframe server. The existing workstations, network, and mainframe were used in the hardware configuration. For software, we used CICS/ESA, DB2, CICS OS/2, DDCS/2, and IBM Extended Services with Database Server for OS/2 to demonstrate their client/server enabling capabilities.

Figure 1 shows the CS92 system configuration.

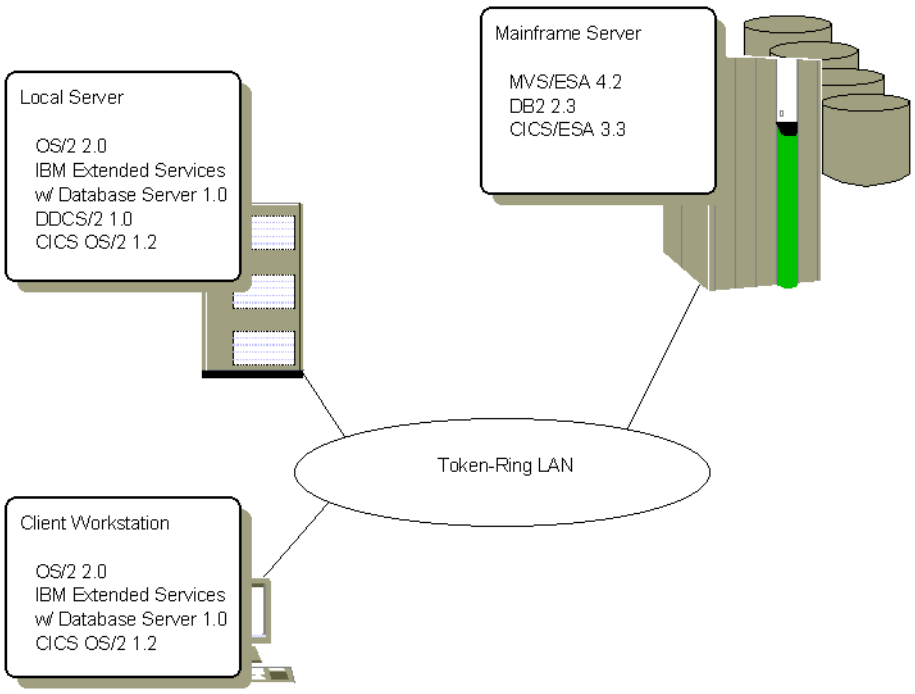



*Figure 1. CS92 System Configuration*

Although we implemented the application in an enterprise LAN environment, using specific hardware and software, the application design using a client/server computing model makes it just as easy to implement the application in other environments, for example, a workgroup LAN environment, using different hardware and software.

### 1.3.1 Hardware Configuration

The workstations on the LAN can be either client machines or local servers. They are connected through a communications controller to the mainframe server, which acts as an enterprise server.

The hardware configuration we used maps nicely to the structure of the bank's business processes described above. The functions provided by the enterprise server could be mapped to the functions provided by the head office. The functions provided locally by the LAN in each branch could be mapped to the functions provided by the branch. Each workstation on the

LAN corresponds to the workstation used by an employee of the branch (for example, a cashier).

### 1.3.2 Software Configuration

We installed CICS/ESA Version 3.3 and DB2 Version 2.3 on the mainframe server running MVS/ESA Version 4.2.

Each workstation had the following software installed:

- OS/2 Version 2.0 with IBM Extended Services with Database Server for OS/2 Version 1.0
- CICS OS/2 Version 1.2

In the project the workstation had the full database manager installed; however, the client part of IBM Extended Services with Database Server for OS/2 would have been sufficient.

The local server machine had installed:

- DDCS/2 Version 1.0 multiuser

The development machines had installed:

- IBM C/2 Version 1.1
- IBM C SET/2 Version 2.0
- EASEL Workbench
- MicroFocus COBOL/2

### 1.3.3 Communications

We used LU 6.2 and NetBIOS as the two communication protocols:

- LU 6.2 was used for the CICS communications between CICS/ESA and CICS OS/2.
- LU 6.2 was also used for the database communications between DB2 and DDCS/2.
- NetBIOS was used for the database communication between the client workstation and the local server.

Figure 2 on page 17 shows the different protocols used by the database and the transaction communications.

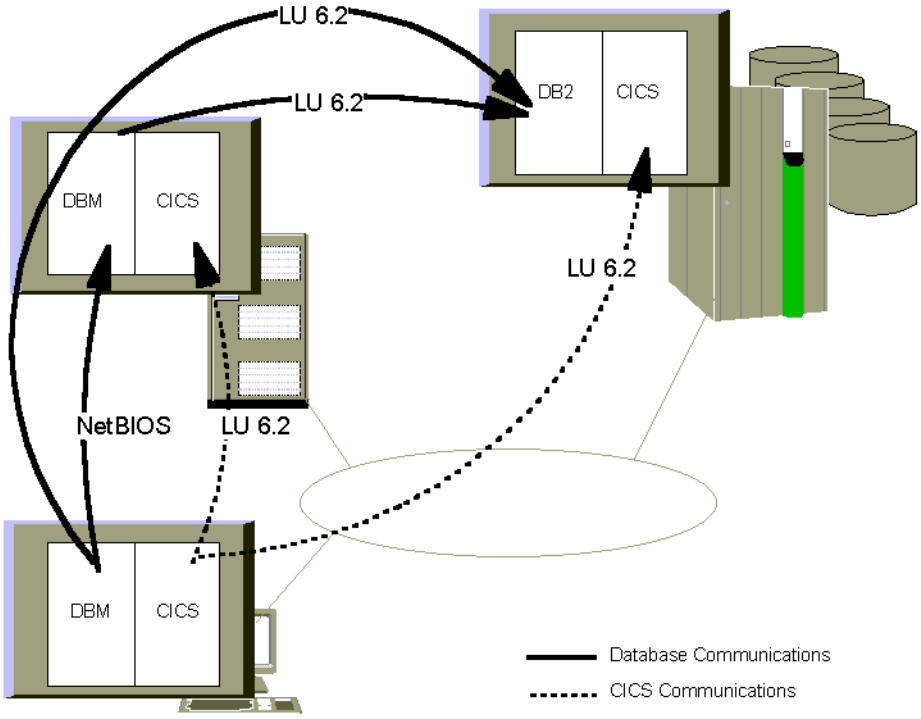



*Figure 2. CS92 Communication Protocols*

## 1.4 Detail Design

The key objective of the CS92 project was to show how the client/server computing model, using an online transaction processing (OLTP) application, could be implemented with IBM's mainline transaction and database products. The project design focused on the following considerations:

***Encapsulation***

All access to data was through application code built around the data. The client application did not use dynamic structured query language (SQL). If the data was resident on the same platform as the client application, data access was built as a separate server application so that client applications would not be responsible for data location or organization.

This encapsulation allowed code to be reused when new server functions were required that resembled functions already developed. The new functions inherited the existing, or base, functions.

### *Transparency*

The project's application design and system design were independent of each other. For instance, services were known by their identifier only, and the service location was controlled independently of the application code. The control of service location introduced some directory services functions into the development.

Although the message routing function was developed at the application level, this layer was considered part of the network operating system layer and was to remain transparent to the application developer.

### *Open Architecture*

Access to the server was through C or COBOL programs only. Any program, not just CICS programs, could invoke a function, written in C or COBOL, that accesses services and data.

The services were developed to be easily portable to other platforms. Because each program was developed using modular programming techniques, the programming language standard set of calls was used and environment-specific calls were not allowed. It should be noted that client functions were not easily ported across platforms because the presentation components were environment dependent.

Any change to server functions would affect only those existing or new client functions that require the function created by the change. There would be no need to amend or relink client applications that required the unchanged or previous version of the server function. This also would apply when the decision was made to change server location.

### *Flexibility*

The decision on how to split functions was made as late as possible in the design phase; minimizing the functions at the client reduces the likelihood of a client process disrupting the system. This thin client approach applied throughout the project.

### *Simple Validation*

Data field validation was applied at the client and server functions: at the client to minimize network and server traffic in the event of simple errors, and at the server so that one server would not become dependent on the many clients correctly performing error handling.

## 1.5 Data and Function Placement

Another important characteristic of CS92 is data and function placement.

### 1.5.1 Data Placement

Data placement was discussed in terms of these considerations:

- Data ownership, which is often the key factor when considering the distribution of data
- Level of sharing, whether by many users or just one
- Data partitioning among the different locations
- Target environment, based on the cost and the software architecture of the target hardware or the security facilities at the location
- Read/change data, including the transaction rates and the data manipulation profile to allow acceptable response times
- Volatility and currency of data
- Integrity, dealing with the coordination of updates in a logical unit of work
- Systems management, such as backup of data, propagation of data, and data recovery

From the above considerations, a decision was made for each entity as to whether to hold a single copy of the data in one location, to hold multiple copies of the data in multiple locations, or to distribute the data, as shown in Table 1.

*Table 1. Table Placement*

| Table | Location |
|---|---|
| Currency | Client workstation, backup on the local server |
| Currency denomination | Client workstation, backup on the local server |
| Exchange rate | Mainframe server |
| Customer order | Local server |
| Customer order denomination | Local server |
| Customer order currency | Local server |
| Cashier | Client workstation and local server |
| Cashier order | Local server |
| Cashier order details | Local server |

| Table | Location |
|---|---|
| Branch order | Mainframe server |
| Branch order details | Local server and mainframe server |
| Customer | Local server and mainframe server |
| Commission | Client machine, local server, and mainframe server |
| Control information | Local server |
| Branch | Local server and mainframe server |

### 1.5.2 Function Placement

The outline design of the application was produced from the requirement specifications. Each business process that was identified was then transformed into application tasks and mapped into application functions.

The decomposition of application tasks into application functions, which can then be implemented as self-contained modules, is key in the design of a client/server application. The application functions had to run on any platform, whether workstation, local server, or mainframe server, and the location of a function had to be transparent to the client part of the application. When the client required a service, it called the function that provides the service but was unaware of where the service was coming from.

In CS92, functions that do not access data were placed close to the user. With functions that access data, the software products used enabled a flexible approach to function placement. Functions that access data need not be influenced by data placement.

The following examples demonstrate the flexibility that was available for function placement:

- The cashier table, which contains the cashier ID and name, was held locally and the server function to get the cashier identity resided on the local workstation.
- The exchange rate table, which contains the buy and sell rate for each currency, resided on the mainframe server, but the function to retrieve the exchange rate data was implemented on the client machine. In this particular case, DDCS/2 enabled the local server function to transparently get the data from DB2 on the mainframe server; no application code was necessary on the mainframe server.

## 1.6 Application Coding

Figure 3 shows the programming languages we used to implement the different components of the application.

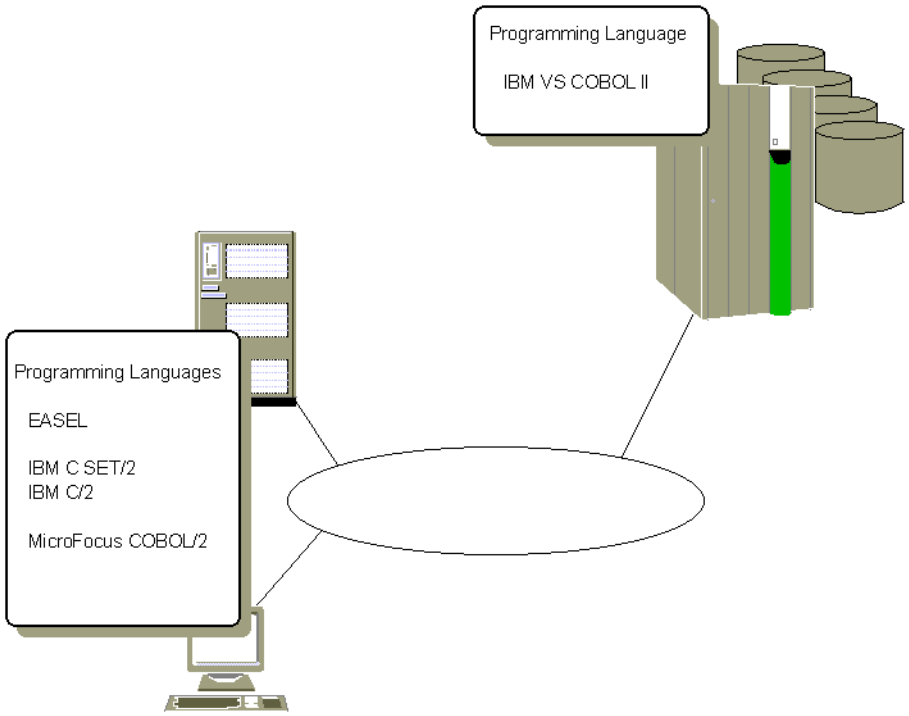



*Figure 3. CS92 Programming Language*

- EASEL was used to implement the presentation component of the application.
- The application component and the application request handler were implemented using IBM C SET/2 (with 32-bit addressing capabilities) to exploit the capabilities of OS/2 Version 2.0.
- IBM C/2 was used to implement the system server interface component to maintain compatibility with CICS OS/2 Version 1.2, which then supported only 16-bit addressing.
- To achieve the portability of the functions that provide the services to the application on multiple platforms, functions on the workstations and mainframe server were implemented in COBOL/2, a programming language used in many business applications. MicroFocus COBOL/2 was

used to implement the functions on the workstation, and IBM VS COBOL II was used on the mainframe server.

## 1.7 Graphical User Interface Design

The approach to designing the graphical user interface (GUI) consisted of the following steps:

1. We identified the processes in the process model that the end user will perform. For example, in order management, the user will perform tasks to carry out such processes as order creation and order update.
2. For each process, we identified the entity in the data model that the user would most naturally use to perform the business process. For example, the customer order entity was selected as the object with which the user interacts in order management. We recommended that a prototype of the GUI be developed together with the end user to confirm that the objects selected for the GUI are correct.
3. For each entity involved, we identified the actions that can be performed on the entity. For example, actions on the customer order entity include:
   - New
   - Update
   - Delete
   - Print
4. We mapped each action of an object as closely as possible to a process. For example, for the customer order object:
   - New action was mapped to the create-order process.
   - Update action was mapped to the update-order process.
   - Delete action was mapped to the delete-order process.
   - Print action was mapped to the print-order process.
5. We identified the data that the user will need to work with on the GUI, beginning with the data for the tasks within a process. For example, in the create-order process, we identified tasks that access customer details, stock details, and input order details. All this information needs to be in the window. The information can be refined during the prototype.

By carrying out the above steps for each object with which the user interacts, we defined the actions and the windows containing data with which the user needs to work. We could then begin the prototype.

# Chapter 2. Network Computing Framework

The Network Computing Framework (NCF) is a skeleton based on the standards of the Open Blueprint. It identifies specific products and services targeted to the delivery and support of network computing.

In this chapter we briefly describe the Open Blueprint and then introduce the NCF. We describe its characteristics and the environment in which to implement it. We also describe the functions implemented in our sample network computing solution.

Open Blueprint is IBM's software architecture that integrates different forms of distributed computing such as network computing, client/server, and mobile computing on a common base. It provides a structure and set of technologies that enable heterogeneous systems to work together and form an integrated system.

In its easiest form, Open Blueprint is a simple chart that acts as a communication vehicle and checklist. In its complete form, it consists of a handbook and technical papers that a software designer would use to make applications on different systems interoperate.

Open Blueprint defines common functional building blocks and interfaces for distributed computing. These building blocks, which consist of different software services, are arranged in logical groups. This building block approach leads to modular, scalable systems that are easy to grow and enhance.

Open Blueprint is the common base architecture on which specific solution-oriented models can be built. The NCF has been developed using the Open Blueprint as the base.

## 2.1 Infrastructure

The NCF is an open, standards-based framework for creating expandable e-business solutions. It is not a product, and it is not based on an IBM-only philosophy (although IBM does offer complete, end-to-end e-business solutions). The NCF is a network computing model containing six key elements:

- An infrastructure and a set of services whose capabilities can be accessed by open, standard protocols and a standard component interface, JavaBeans

- Clients based on a Web browser Java applet model that support universal access, a thin client paradigm, and exploitation of just-in-time delivery of components to provide a rich user interaction with the server
- A programming model and tools that create and exploit JavaBean components. As a result, any tool can produce a component to access any service
- Internet-ready protocol support, such as Hypertext Transfer Protocol (HTTP) and Internet inter-ORB Protocol (IIOP), which links JavaBean components
- A set of connector services that provide access to existing data, applications, and external services
- A set of built-in collaboration, commerce, and content services as a foundation for an industry of partner-built solutions and customizable applications for e-business

Figure 4 shows the NCF infrastructure.

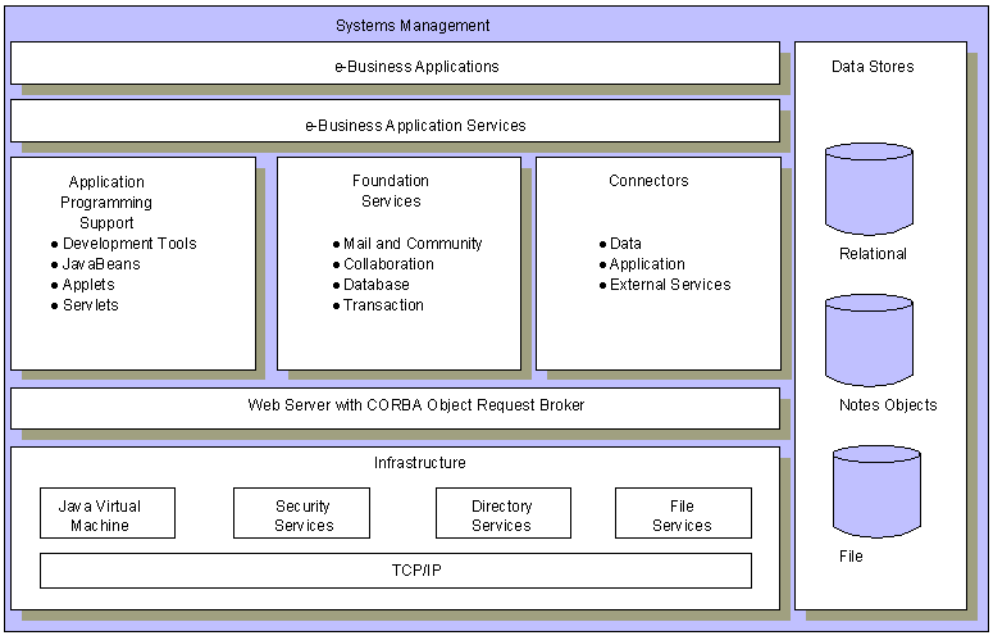



*Figure 4. Network Computing Framework Infrastructure*

## 2.2 Components

The NCF has eight majors components, each of which is discussed in detail in the following subsections:

- Clients
- e-business application services
- Data and transaction connectors
- Application programming support
- Foundation services
- Web server with Object Request Broker (ORB)
- Infrastructure with Java, directory, and security
- Systems management

### 2.2.1 Clients

The client element of the NCF provides universal access, a thin client paradigm, and exploitation of just-in-time delivery of components for seamless interaction between a user and information provided by the server. New NCF applications can be written as hypertext markup language (HTML), including dynamic HTML, with Java applet extensions. Because the client model is based on standard browsing protocols (HTTP) and supports additional standard protocols such as IIOP, Lightweight Directory Access Protocol (LDAP), Post Office Protocol (POP), and NetNews Transfer Protocol (NNTP), it offers access to key information and applications.

The NCF client model provides benefits in several key areas:

- A browser technology, used for distributing information and downloading applets from the network, making it easy to deploy, use, and manage new solutions. The NCF also provides several other client options based on user functionality requirements:
  - A simple browser with Java applet capability, so that Java applets can use the ORB and IIOP to communicate with remote objects
  - Domain-specific desktops that use dynamic HTML and Java applets to integrate local client applications with Web server access to the network
  - Application-specific clients, such as e-mail
  - A fully integrated client such as Lotus Notes, which provides rich collaboration and superior mobile support

- A consistent programming model, which allows developers using Java to learn once and write anywhere. Having a consistent programming model among clients means that end users can reap the unique benefits of those devices while taking advantage of common Java and browser-enabled applications.
- Dynamic data exchange, implemented as JavaBean components. JavaBeans can dynamically exchange data, using the Lotus InfoBus technology provided in the JavaSoft Java Development Kit (JDK).
- Support of mobile and fixed clients, from personal data assistants to network computers (NCs) and from personal computers (PCs) to high-end workstations. The variety of mobile clients supported by the NCF operate either in disconnected environments or connected environments, using wireless and other low bandwidth connections. These clients are supported through the use of replication technology. NCF mobile communication is enhanced with the use of compression, caching, and differencing algorithms.

### 2.2.2 e-Business Application Services

NCF e-business application services are building blocks that facilitate the creation of e-business solutions. They are higher-level, application-oriented components that conform to the NCF programming model. They build on and extend the underlying NCF infrastructure and foundation services with functions required for specific types of applications. As a result, e-business solutions can be developed fast and with high quality.

One set of these building blocks is being defined to support the process steps used in electronic commerce solutions. Intellectual property management and secure business-to-business delivery of content will be supported as well as the secure electronic transaction (SET) standards developed by VISA and MasterCard with IBM and other leading companies. Additional e-business application services will be defined over time.

### 2.2.3 Data and Transaction Connectors

Any application that operates in support of an enterprise must access information and business processes on a variety of computers. The bulk of today's critical data and application programs resides on and uses existing enterprise systems. Hence both developers and operations teams need a tool set that facilitates connecting existing applications to the new applications. NCF connectors enable you to extend your critical corporate data, application, and transaction programs to the Web, linking the best of the Internet with the best of the enterprise:

- IBM CICS Gateway for Java brings open access to CICS from any Java-enabled Web client, such as a Web browser or a network computer. This Java applet can directly call CICS programs and data and run on any server platform that is Java-enabled.
- IBM CICS Internet Gateway provides an interface between a Web server and a CICS application using common gateway interface (CGI) scripts. It allows the conversion of 3270 data streams to HTML format.
- IBM Net.Data allows Web developers to use macros to easily build dynamic Internet applications. Net.Data Web Macros have the simplicity of HTML with the power of dynamic SQL. Net.Data provides database connectivity to a variety of data sources, including information stored in relational databases and flat files. Your data sources, such as DB2, Oracle, and Sybase, can be on a wide range of operating systems.
- Lotus connectors integrate the Domino server with a broad range of database, transaction, and enterprise application systems. The Lotus connectors enable Lotus Notes clients and Web browsers to access enterprise data and applications, including enterprise resource planning systems, such as SAP R/3, traditional transaction processing systems, such as CICS and IMS, and relational database systems including IBM DB2 and offerings from Oracle and Informix.
- IBM IMS Internet Solutions contain four solutions that provide connectivity from the Web to transactions in IMS databases. These solutions enable customers to match their communication infrastructures used with IMS to their Web server configurations.
- IBM MQSeries Client for Java enables Web browsers and Java applets to issue queries over the Internet to MQSeries for information stored in mainframe and legacy applications.
- IBM MQSeries Internet Gateway provides a transparent bridge between the Web and MQSeries commercial-messaging applications.
- IBM Host On-Demand provides fast and easy access to mainframe data from the Internet and intranets. Host On-Demand is a high-performance, low-cost solution for Internet users.
- DCE Encina Lightweight (DE-Light) Client, combined with an intermediate gateway, provides a solution for secure network transactions in distributed environments. DE-Light uses Kerberos and Secure Socket Layer (SSL) security to support the integrity of information being processed in a transaction from a Web browser to the server.

### 2.2.4 Application Programming Support

Two concepts are essential to understanding IBM's approach to NCF application programming support: JavaBeans components and the assembly of those components.

JavaBeans components are the core of the delivery model, for both client-side application support and server-side application programming. Thus you can take advantage of the development productivity provided by Java's reusability of objects. JavaBeans is the organizing principle of the NCF programming model. It is the platform-neutral, component architecture for Java to develop or assemble network-aware solutions. The JavaBean component model specifies how to build reusable software components, how the resulting JavaBeans describe their properties to visual tools for rapid application development, and how the JavaBeans communicate with each other.

Here are a few examples of JavaBean components:

- JavaBeans that provide easy access to relational and hierarchical data
- JavaBeans that integrate the existing transactional applications (CICS, IMS, and Encina) of a business into new Web-based applications built within the NCF
- JavaBeans to access applications from vendors such as SAP, Baan, and PeopleSoft
- JavaBeans that integrate all of the Domino server classes into the NCF programming environment
- Lotus JavaBeans for embedding functions such as text processing, spreadsheets, charting, and calendaring into applications

JavaBeans enables developers to create reusable software components that can then be assembled together with visual application builder tools, such as VisualAge for Java. In the NCF, application development tools are divided into three categories, summarized below:

- Content authoring tools, used to create and manipulate the Web-based application's multimedia content including HTML, graphics, images, animations, audio, and video
- Content assembly and management tools, taking the active contents and composing and scripting them together with other active or static contents to create a complete NCF application. Thus anyone with neither sophisticated content development skills nor application development skills can create the final product.

- Integrated development environment (IDE) tools, including tools to support component coding and development such as Lotus Notes Designer for Domino or VisualAge for Java

### 2.2.5 Foundation Services

The foundation services are a subset of the Open Blueprint Application Services resource managers. They are products that provide essential services to e-business applications, but they are not intrinsically e-business applications themselves. The services are:

- Mail and community services, which provide e-mail messaging, calendaring and group scheduling, chat, and newsgroup discussions. These services support standard Internet technologies and protocols and are accessible from all standards-based clients.
- Collaboration services, which provide groupware and workflow support. They integrate information from any resources, manage how much and what is shown to the user, allow users to interact with the information, and manage the content.
- Relational database services are delivered through IBM's Universal DB2. They provide a facility for managing both the operational data and multimedia content. They leverage existing business logic to build new Web-based applications and using Java support development and execution of database application programs, stored procedures, and user-defined functions. They support the SQL and the Java database connectivity (JDBC) interfaces and integrate transaction services through the XA model.
- Transaction services extend the Web by providing a transactional application execution environment. The services support development of both procedural and object-oriented transactional application programs.

#### *Domino: A Foundation Service*

Domino provides the basic services to deploy a Web site and a network computing application environment. It includes a Web server and the services required to run Java servlets. It supports the runtime environment for the NCF programming model and incorporates infrastructure that uses standard LDAP and SSL. Domino offers e-mail, newsgroups, chat, and calendaring and scheduling—all based on standard Internet protocols. Its goal is to extend the reach of a business's support for its customers, partners, and employees. It provides a robust implementation of the directory and security services based on the Domino object store, while adhering to the standard LDAP, X.509v3, and SSL. With Domino, developers have the option of exploiting the Domino document object store for managing their Web content.

Finally, Domino offers rich, mature workflow and collaboration solutions. The goal is to reduce product cycle times, support ad hoc teams, and enhance the efficiency of the enterprise. Domino supports a prefabricated application framework and set of modules to speed the deployment of business applications. These include Domino.Action, Domino.Merchant, Domino.Doc, and Domino.Broadcast.

### 2.2.6 Web Server with Object Request Broker

Just as Java and component-based assembly are central to the application development strategy for the framework, so is the Web server a powerful point of integration in the NCF. Electronic networking businesses today presuppose the use of the Web and therefore a Web server.

The Web server is the entry point to server functionality supporting HTTP and IIOP requests from NCF clients, locating and invoking business logic on the logical midtier server. The Web server provides the programming environment for transactional business applications, linking the Web to the existing enterprise applications, data, and systems.

### 2.2.7 Infrastructure with Java, Directory, and Security

The basic infrastructure services are Java, directory, and security.

For both clients and server, the Java Virtual Machine provides the base support for programming. It also provides platform independence for e-business solutions.

Directory services that locate users, services, and resources in the network are accessed through standard LDAP.

Security services support user identification and authentication, single signon, access control to resources and services, firewalls, confidentiality, data integrity, nonrepudiation of transactions, security management, key recovery, and Java security. They are based on robust public key support and certificates.

### 2.2.8 Systems Management

Finally, there is the fundamental assurance that all of the NCF components can be managed. NCF systems management encompasses systems, network, and application management. The NCF provides for the unique management requirements of network computing across all elements of the system including applications, services, infrastructure, and hardware. The basic systems management model has four basic categories:

**Development:** The IBM products included as part of the NCF already have the necessary interfaces to facilitate management. Part of the NCF is to provide toolkits that enable customer-written applications and services to take advantage of the same management services.

**Deployment:** Once applications have been developed, it is essential to ease the process through which they are delivered to the Internet servers.

**Operations:** This aspect of systems management provides support to the Webmaster for statistics, performance tuning, and load balancing, along with more traditional backup, configuration management, and security services.

**Content:** These tools provide support for the visualization and management of the objects, links, and files that constitute e-business on the Web.

# Chapter 3. From Client/Server to Network Computing

In this chapter we explain the evolution of the computing model. We start from the characteristics of the client/server application. Then we analyze their impact, or their modification, in a network computing environment. We also describe the benefits the new model can provide. Finally, we discuss how the client/server application building blocks can be used as is or modified in a network computing application.

## 3.1 The Client/Server Model

The client/server computing model for distributing applications is a logical model that defines the existence of a client (requester of a service) and a server (provider of the service). A service can entail data, a function, or any resource that can be shared. The server can be local or remote. Usually, the server is remote, and there is thus a need for communications to connect the client and server (or servers).

The evolution from the client/server model to the network computing model brought with it a change in the communication protocol used and the definition of the client:

- Today, TCP/IP provides the ability for corporations to merge different physical networks while giving users a common suite of functions. It allows interoperability between equipment supplied by multiple vendors on multiple platforms, and it provides access to the Internet. In fact, the Internet, which has become the largest computer network in the world, is based on the TCP/IP suite.

  Unlike the Internet, the intranet has evolved recently, consisting of TCP/IP networks that are entirely under the control of a private authority or company. Those intranets may or may not have connections to other independent intranets (which would then be referred to as *extranets*) or the Internet. They may or may not be visible from the outside depending on the implementation.

- The thin client, as presented by NCs, offers a range of simple, economical alternatives to terminals and underutilized PCs. NCs extend access to network applications, intranets, and the Internet while they lower the total cost of ownership with reduced up-front expenses and significantly less need for support. They are simple to install and easy to use and manage. With the "install once and run everywhere" nature of network computing, including the growing availability of Java applications, NCs make the business desktop easier to manage and support.

With NCs, there are no longer any local services on the client machine. There is no requirement for any storage on the client machine: The application is dynamically downloaded at execution time, and data is not stored on the client machine. However, to support specialized users, such as mobile users, a client with storage and local services capabilities is an effective alternative. In a Domino environment, Lotus Notes clients can run applications disconnected from the Domino server, requesting only a connection periodically.

### 3.1.1 Distributed Computing

The client/server computing model has evolved to a distributed network computing model, which can be viewed as a single multiuser computing system that functions and performs like a traditional host computer but is built from a set of discrete machines, interconnected by a very high-speed, reliable mechanism. The whole structure is transparent to the application. Services can easily be assigned to server machines on the basis of the changing requirements of users and the changing capabilities of server machines.

In general hardware terms, the application uses workstations or NCs, network connected servers, and enterprise mainframe servers. Each runs on different platforms but all are interoperable because they are linked by the network, thus maximizing the processing power of each machine. In the network computing model, the concept of distributed computing is extended further, as applications can be distributed to any client machine that has a Web browser, without any preconfiguring of the network.

### 3.1.2 Transparency

When CS92 was developed, a significant characteristic of the client/server model was that it provided transparency to applications and end users, leaving them untroubled by the complexity of the overall system. Each part of the system encapsulated its own complexity and communicated with the remainder of the system and users by presenting a simple view.

The following features have transparency considerations:

- User view

  The CS92 user saw the application as a single, coherent application that provided the means of performing the business functions. The user did not need to know how or where a function or resource was provided.

  User view transparency is the same in network computing: a successful migration should mean that the user is unaware of the change. This is one major requirement for our migration project.

- Location

  The CS92 user was be able to access applications and data without regard to their location in the distributed system. Programming interfaces for accessing data and services enabled requesting programs to be independent of the location of data or services.

  This independence is particularly true of the client in the network computing model: The client can be located anywhere on the enterprise network or even connected through the Internet (if there is no security exposure) and still be able to access the application. Furthermore, the client can be any kind of hardware, from NCs to PCs, because the server supports most of the load.

- Vendors

  Both client/server and network computing enable machines from different vendors to interoperate. Network computing simplifies the interoperability as there are no specific hardware requirements for the client machine.

- Network independence transport

  When CS92 was developed, many different networks had to be crossed to access data and applications. This diversity of networks meant that applications had to use communication interfaces that insulated them from the protocols of the particular network transport they used.

  In our network computing migration, we standardized on one network protocol, TCP/IP, which facilitates the communication among the servers, the subsystems such as Domino, DB2, or CICS, and the programming of the Domino application.

- Distributed applications and service enablers

  A basic requirement in both client/server and network computing is that the application developer be able to concentrate on the implementation of a function, rather than on how to make the function accessible to requesters anywhere in the distributed system.

  We satisfied this requirement in our migration because we used Domino to create and run the application. Therefore the application was downloaded dynamically, on the user's computer—a Lotus Notes client or a Web browser—at execution time. The enterprise gateways, in our case, the Lotus connectors, enabled clients to connect to the enterprise services.

- Performance

  Clearly, every application requires adequate performance. In CS92, it was anticipated that when function and data were accessed directly from the

workstation, performance would be improved. However, no performance data was gathered.

In the case of network computing, all accessing of function and data is by way of the network. It is not yet clear whether the costs of the network times will be outweighed by the benefits of the powerful mainframe processors, improving performance time. It is beyond the scope of this book to analyze performance statistics.

- Universal distributable directory

  A directory is required that provides universal naming and naming consistency so that information about resources and facilities in a distributed environment can be managed and located. That is, code on the client does not need to know the identity and location of the server it is making requests of, but some facility does need to know. In the foreign currency application, code was written on the client machine to satisfy the requirement. However the developers acknowledged that this was not a full solution.

  Within Domino the whole directory is stored in a single integrated and distributable database that also contains the other application objects. Therefore the directory itself is programmable and may be part of the application.

- Data conversion

  In a centralized environment, the entire application is contained in one system and there is only one data representation on that system. For an IBM mainframe, the data representation is EBCDIC, and for a workstation, it is ASCII.

  As soon as data is exchanged between systems, however, it is important to know whether the systems that exchange data have the same type of data representation.

  If not, data conversion is required. Several approaches to data conversion exist in the industry. One approach is to convert data to a common known representation. Another is to convert incoming data to the representation known by the local system.

  Whatever approach is taken, the application should be shielded from the data conversion. The conversion of the data should be taken care of, for instance, by a transaction manager or resource manager so that neither the conversion nor the data representation will affect the application.

  CS92 used the CICS macro DFHCNV to convert data when it arrived at the mainframe.

In the case of our migration to a network computing solution, the data conversion was performed by the Lotus connector code, thus making it transparent to the application code.

## 3.2 Network Computing Benefits

CS92's developers described some benefits expected to be achieved through developing a client/server solution. Here we discuss those benefits in terms of what client/server computing provided and what network computing can provide:

- Cost-effective solution (right-sizing)

  When the original solution was developed, it was sometimes thought that the best way to right-size was to place a significant proportion of the processing on the inexpensive workstation.

  For certain types of applications, the current thinking is that a right-sized application has most of its processing on the mainframe, because a cost-effective solution is made up not only of the capital costs of the equipment but also the operating costs of the system. Network computing follows this philosophy.

- Provides GUIs for users

  The use of workstations is a cost-effective way of providing a GUI to users. Both client/server and network computing applications can be designed such that the user interface processing is performed on the workstation. Well-designed GUIs for appropriate applications provide significantly higher end-user productivity than those that run on traditional 3270 screens.

  A network computing application can also build a dialog using a Lotus Notes form, HTML on a Web page, or a Java applet. Developers have the choice of building a GUI dialog or an HTML dialog, depending on which is more appropriate for the application.

- Organization structure mapping

  In CS92, the bank consists of a hierarchy of semi-independent units that share common information, for example, commission rate. The lowest units in the hierarchy are the cashiers who have their own data, such as personal stock levels. The cashiers require services unique to them such as creating new customer orders. At the same time, they also require data and services held higher in the hierarchy: exchange rate is an example of this. Dealers maintain the exchange rate at the central office because such data is dynamic and the latest rate must always be applied.

The branch office is in the middle of the hierarchy. It provides services as a response to the cashiers' requests to provide, for example, replenished stock. In turn, however, it acts as a client in relation to the central office, for instance, by making requests to return excess currency.

The head office is at the top of the hierarchy. It provides services to both the branch and the cashier. The three levels of the bank illustrate the client/server model in which a client is a component that requests services, and a server is a component that satisfies requests. The example of the branch office shows that a server can also be a client of another server. The hierarchical structure is different from peer-to-peer structures and maps more logically to the organization of a business.

The network computing framework maps more closely to the way in which modern business is evolving. The trend is for one enterprise's computing system to communicate directly with that of another enterprise: the connection is made without any regard to the organization of either business. In a similar way, when individual customers browse an enterprise's home page, and possibly order goods or services, they do not care which department or level of the organization is dealing with their order.

Domino collaboration features also support the business requirements. It allows organizations to set up powerful connections among them while maintaining control without enforcing a hierarchy.

- Exploits LAN infrastructure

  When CS92 was developed, many enterprises had a LAN infrastructure with many workstations installed. Client/server applications can successfully use such a structure.

  This usage is not such an important consideration for network computing. Provided that a workstation can communicate with its server machine, using TCP/IP, it is not significant whether a LAN infrastructure is used or not.

- Improves availability

  The infrastructure of a client/server application can be used to improve availability. For example, if availability is crucial to the application and sufficient resources are available, several servers can be defined to perform the same functions. Single points of failure are avoided by running these servers on different machines. When only one server provides a particular function, the function can be switched transparently to an alternative server if the first one fails.

Domino network computing applications and data can be replicated automatically on the Lotus Notes client workstation, providing users with a disconnected application whether they are mobile or the server is unavailable. Web browser applications are rather more dependent on the network: If the network goes down, they cannot run. However, networks are becoming increasingly more robust.

- Enables heterogeneous interoperability

  Client/server applications are based on accepted industry standards and allow for heterogeneous interoperability, so that systems from different vendors can work together to execute applications.

  Java is evolving into the industry standard in which to write network computing applications. This evolution is based on popularity and actual usage partly because of Java's ability to "be written once, run anywhere."

  Domino applications are part of a Lotus Notes database that can be stored on any hardware supporting Domino (S/390, AS/400, UNIX, Intel). The applications can also run on any Lotus Notes clients or Web browsers.

## 3.3 Building a Client/Server Application

In this section we describe some features of client/server applications and discuss some considerations to take into account when you build such applications. We also discuss the applicability of these characteristics to a network computing application.

### 3.3.1 Basic Communication Models

Three communication models can be implemented, with varying degrees of success, by client/server or network computing:

- Conversational model

  In the conversational model, a conversation is a series of synchronous message exchanges between two partners that have a peer-to-peer relationship. Both partners must be active for the duration of the conversation. Either partner can initiate and direct the communications, and both sides can send or receive data. In general, a receiver cannot send data until the sender surrenders that right.

  The conversational model is powerful but may make the programs more complicated. Both sides have to know the logic of the conversation, and they have to agree on the state of the conversation at any given time. From a programming point of view, the two sides of the conversation are not transparent to each other, although they may be to the end user.

This model is difficult to implement in a network computing application because the server is generally unaware of the state of the browser, especially when HTML is used. However, there have been some enhancements to the middleware, such as CICS, to improve this conversation support.

- Call model

  In the call model, a program sends a request to another system by specifying a program or function to be executed. The process is synchronous to the requester, which waits for the function to complete. The called system executes the requested program or function and returns the results in a predefined format to the calling program. The call format includes parameters for the passing of data.

  The routing of the request and the establishment of communication with the called program are transparent to the caller.

- Messaging model

  In the messaging model, a single message is passed from one program to another by placing the message on a queue. No response is required from the client. The message is not necessarily processed instantly. There can be a single queue or multiple queues for different message types and priorities.

Our migrated application uses either the call model or messaging model, depending on the configuration:

- The call model is used for Web and Lotus Notes users, when all participants—servers and networks—are available. It is also the model implemented by TCP/IP.
- The messaging model is used to support mobile users as a connection is not always available.

### 3.3.2 Application Characteristics

When we migrated the application from the client/server model to the network computing model, we had to consider the application characteristics to ensure that the new model maintains or improves them.

In CS92, an OLTP was critical to the business of the bank. Hence the following requirements had to be satisfied:

- Rapid response times
- High availability
- High volume transaction rates

- Robust data integrity

### 3.3.3 Security

Security is a critical issue in any application. Only authorized users should be able to access applications and data. In centralized applications, access control to resources is often managed by a resource access control manager through user IDs and passwords. Preferably, only one resource manager should exist for maintaining and checking access control.

When applications and data are distributed over more than one system, the task of securing applications and data becomes more complex. In the first place, more than one security resource manager is involved, and heterogeneous environments can involve different resource managers. These different resource managers must be able to interface with each other. In the second place, the maintenance of user IDs and passwords has also become more complex in a distributed environment.

Security can be divided into three areas:

- Authentication

  When clients communicate with servers, they must identify themselves, and the server must verify that the clients are who they claim to be. The authentication function should be contained either in the security subsystem or in the communications subsystem. Several solutions related to authentication exist in the industry.

  Domino authenticates Lotus Notes users through certificates based on public and private keys. It authenticates Web users through an entry in the Domino server Names and Address book and possibly an HTTP password (see “Lotus Notes Client” on page 76 and “Web Client” on page 76).

- Authorization

  After authentication has taken place, it is necessary to check whether the client is authorized to access the resources it is requesting. The authorization check is done through the use of user IDs and passwords. First, the user ID and password are used on the client machine to check whether an end user is authorized to use the client application. Then, the client application asks for services in the network, and the user ID and password are sent to determine whether the client is allowed to access those services.

  Domino provides additional security mechanisms to protect its resources. Using access control lists, you can protect databases, forms and views, and documents and fields (see “Server Security” on page 75).

- Encryption

  To secure the transmission of data through the network, data should be encrypted. Again, the encryption function should be carried out by the communications subsystem or the security subsystem. The application programmer should not be bothered with this function. Domino offers encryption for both Lotus Notes and Web users (see *"Encryption" on page 76*).

### 3.3.4 System Management

System management is crucial for effective operation of any application. There are particular considerations when the application uses a client/server model. Some of these considerations are simplified when the application uses network computing. Below we discuss some of these considerations:

- Network

  The network is essential for client/server or network computing applications. The system management tasks include ensuring the continuous availability of the network, and acceptable performance of the network, in both response time and throughput. For client/server and intranet solutions, usage of, and access to, the network must also be managed.

  A network computing application may also need to handle more throughput, especially in the initialization phase of the application, when the form is being loaded from the server or when the entire database is being replicated to the Lotus Notes workstation.

- Software distribution

  Software distribution is complex in a client/server environment. Version control between systems is required to ensure software integrity. Rollout must be coordinated to ensure that the application comes online consistently across the enterprise and that the rollout has no impact on the user.

  To distribute the new or updated software, system administrators define packages that contain data files, programs, system software, and script. They then define the distribution jobs by specifying package, site, execution time, and query. Installation is then by script driver, triggered by user logon.

  Software distribution becomes much more straightforward with Domino. Its effective replication solution allows the replication of a database containing the application and its associated data from the Domino server

to the Lotus Notes client. In a case of a Web server, only the Web browser code needs to be loaded at the client.

- Problems and changes

  Problems do occur in applications, and they need to be reported, investigated, fixed, installed, and tracked throughout their lives. In a client/server environment, users can easily report a problem to their support team. The support team, however, may have much greater difficulty in analyzing and reproducing the problem, as they are unlikely to have an identical environment in which to run the application. When a fix, or a change, is made to a client/server application, the installation faces the difficulties of software distribution.

  In an intranet network computing environment, users can still report problems to their support team, but the team's investigations should be much simpler as there is only one environment and one level of code to analyze. Similarly, the installation of fixes and changes needs to be done in only one place.

  The mechanics of problem management are the same in an Internet network computing environment. However, culturally, the nature of problems may change, leading to discussions over the following questions:

  - Will an external user make the effort to report a problem, particularly one of low severity?
  - How will the external user report a problem?
  - Do external users have to prove their credibility?
  - What will persuade a company to fix perceived problems?

### 3.3.5 Data Management

The tasks required to manage databases include reorganizing the data, backing it up, control, performance monitoring and tuning, capacity planning, housekeeping, and access control.

The operational success of an application depends in part on the accuracy, consistency, and completeness of its data. Here we look at the subjects of backup and recovery and data integrity in relation to client/server and network computing applications.

- Backup and recovery

  In a centralized environment, backup procedures are often implemented and maintained automatically. In a client/server environment, the backup of data on client machines cannot easily be done in this way. Users are unlikely to back up the data on their own machine in a systematic, disciplined way, and it cannot even be guaranteed that users will either

always turn off their machines or always leave them on to enable remote access.

In network computing, the databases are no longer distributed to client machines, so the required data management tasks do not have to be replicated across remote environments. Backup and recovery can again be centralized, because no data is stored on the client machines.

- Data integrity

  Data must be kept in a consistent state, that is, it must have integrity. The developers of client/server applications were unable to ensure such consistency, because one business transaction was implemented by two logical units of work, one updating data on the client machine, the other updating data on the server machine. At the time, the two-phase commit protocol was not implemented by system software across different platforms, which would have ensured that either both updates were successful or neither was. The problem could not be solved by combining the two updates into one logical unit of work because the level of Distributed Relational Database Architecture (DRDA) used did not support updates across platforms.

  With network computing, data is typically held on only one platform, so there is no longer a problem with cross-platform updates, and normal application design should determine what constitutes a logical unit of work.

# Chapter 4. Network Computing Environment

In this chapter we describe the new network computing environment to which we migrated our client/server application. We concentrate on the products and features that we used in the migration:

- Domino and Lotus Notes
- Lotus connectors

## 4.1 Domino and Lotus Notes

Domino is a software solution that combines three essential technologies: a reliable and innovative client/server messaging system, groupware, and the Internet. Domino gives you the full-function power to create custom applications for improving the quality of your everyday business processes. It includes an application development environment, system administration capabilities, Lotus Notes desktop, and mail functionality.

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. With Domino, you can rapidly build, deploy, and manage applications that help coworkers, partners, and customers collaborate and coordinate critical business activities online. Domino supports a variety of clients and devices, including Web browsers, Lotus Notes clients, and POP3 and Internet Message Access Protocol 4 (IMAP4) mail clients.

Domino is an Internet applications server. It combines the open networking environment of Internet standards and protocols with the powerful application development facilities of Lotus Notes, enabling you to develop a broad range of business applications for the Internet and intranet.

Domino can be thought of as:

- A server that can simultaneously act as a Web server and a traditional Lotus Notes server
- A development environment for building Web and Lotus Notes applications
- An enabler for both the Web browser and the Lotus Notes client

### 4.1.1 Domino Services

Domino provides a set of integrated services to build, deploy, and maintain secure, interactive applications on the Internet, corporate intranets, and extranets. These services include:

- Programmable object store

  The container for data (a collection of documents and the application that manages these documents) within Domino is a Lotus Notes database, sometimes referred to as an *object store* or a *repository*. Domino features a programmable object store that is optimized to store, manage, and retrieve complex data types as well as forms.

- Information replication

  One of the key features of Domino is that you can distribute copies of Lotus Notes databases to clients (thus supporting mobile users) and other servers. Bidirectional replication automatically distributes and synchronizes information and applications across geographically dispersed sites. Replication makes your business application available to users around your company or around the world, regardless of time or location.

- Messaging services

  An advanced client/server messaging system with built-in calendaring and scheduling enables individuals and groups to send and share information easily. Message transfer agents (MTAs) seamlessly extend the system to SMTP/MIME, x.400, and cc:Mail messaging environments. The Domino messaging service provides a single server supporting a variety of mail clients —POP3, IMAP4, Messaging Application Programming Interface (MAPI), and Lotus Notes clients.

- Enterprise integration

  The Lotus connectors seamlessly link Domino with relational databases transaction systems and enterprise resource planning systems, through real-time and batch-level bidirectional integration, enabling you to interconnect your people, integrate your information systems, and improve your business processes.

- Security and Rivest-Shamir-Adleman (RSA) authentication

  Based on RSA encryption, the Domino security model provides user authentication, digital signatures, flexible access control, and encryption. Domino's security enables you to extend your intranet applications to customers and business partners.

- Directory

  A single directory manages all resource directory information for server and network configuration, application management, and security. Domino includes user account synchronization between Windows NT and Domino and is LDAP compliant. The directory is the foundation for easily managing and securing your Internet and intranet applications.

- Workflow

  A workflow engine distributes, routes, and tracks documents according to a process defined in your application. Workflow enables you to coordinate and streamline critical business activities across your organization, and with customers, partners, and suppliers.

  The combination of messaging with calendaring and scheduling functionality (handling any information related to a date and time) provides you with the tools to create workflow applications.

- Agents

  Agents enable you to automate frequently performed processes, eliminating tedious administration tasks and speeding your business application. Agents can be triggered by time or events in a business application.

- Powerful development

  Lotus Notes Designer for Domino, available separately, features an integrated development environment that gives developers and designers hassle-free access to all the features of the Domino server. Lotus Notes Designer for Domino offers enhancements that make creating advanced Web sites fast and easy. It also includes two valuable new tools: Lotus BeanMachine for Java, and Lotus Notes Global Designer. Use BeanMachine to create multimedia Java applets in minutes without writing any Java code. With Lotus Notes Global Designer, your applications can run in many different languages at once.

- Network and mobile support

  Domino supports a vast variety of network operating systems and protocols. Based on replication, it provides mobile users with seamless access to the data of the databases stored on servers and clients.

### 4.1.2 Server Features

In this section we describe the Domino features that are of interest when you migrate your application to a network computing application.

#### 4.1.2.1 Application Perspective

Domino is a true client/server application. Therefore its use in a migration from client/server to network computing does not force a change of paradigms but enables you to build cross-platform solutions and preserve today's investments for the future. Domino supports a wide range of operating and hardware platforms. Applications built on one platform can be hosted on another with no recompilation.

Domino integrates security, messaging, workflow, replication, and application development technology in a single platform. It allows you to add communication, collaboration, and coordination support to your business logic.

Using its programmable object store, you can easily add mixed data and functionality to your application, such as rich text, images, application objects, and multimedia (for example, audio, video). You can combine a traditional business transaction with meaningful images and multimedia to create an attractive new look and feel application.

In a Domino application, specific actions can be triggered according to the value of a specific field or the status of the document itself, allowing the automatization of tasks. For example, you can create an activity based on a due-to-date field value and the status of a check document to send an e-mail reminding the customer of the outstanding payment.

The Domino object store provides cross-database, cross-server, full-text searching, thus enabling users to perform full-text queries across multiple databases on multiple servers.

Domino supports mobile users, who most of the time work in a disconnected mode. Its replication feature allows synchronization of the data of the mobile user workstation with the application's data on the server, whenever the mobile user establishes a session with the server.

#### 4.1.2.2 Web Perspective

As a Web application server, Domino supports a variety of Internet clients: Web browsers, POP3 mail clients (such as cc:Mail or MS Mail), and news readers. Unlike other Web servers, Domino also supports a wide variety of other client types based on non-Internet standards, such as MAPI-based mail clients, telephones, pagers. And, of course, Domino fully supports the Lotus Notes client.

Domino offers new directions for interenterprise connectivity by extending your company's supply chain, from suppliers and business partners all the

way to customers. Domino's replication mechanism can, for example, ensure the secure delivery of information to business partners through the Internet by replicating only the necessary subset of information to a Domino server that is located outside your organization's firewall. This selective bidirectional replication is more powerful than simple mirroring, which allows you to copy all of the data from one server to another.

You can manage the activity that is driven by the information posted on the Web site, using the communication and collaboration features Domino offers. Domino's group calendaring and scheduling functions enable you to implement workflow logic for business processes that have time-sensitive actions and tasks associated with multiple users.

#### 4.1.2.3 Management and Security Perspective

Domino uses the replication process to distribute application or design element changes. Therefore, Domino offers a seamless cross-platform deployment capability. Document-related changes are implemented fast, because a change in a form is inherited by all documents that use that form. Furthermore the use of templates enables you to automatically distribute changes within your organization. Therefore Domino contributes significantly to reduce your system management efforts.

As most security systems on the Web are designed to keep almost everyone out or to let everyone in, Domino offers a wide range of effective security services, to establish role-based access to content. The granularity of Domino's security services together with its directory service enable you to establish multiple, role-based layers of security for both Web users and intranet users. This is a strong argument for choosing Domino if one of the objectives of your migration is to reach a wider audience.

### 4.1.3 Development Features

In this section we describe the benefits of using Lotus Notes Designer features to migrate your application to a network computing application.

#### 4.1.3.1 Rapid Application Development

Domino application development technology supports rapid and effective development and deployment of the application. It also provides different application programming interfaces (APIs) to enable your developers to use the most appropriate product for the task at hand.

Domino ships with a complete set of templates for the most common types of applications. These templates include all of the logic and formatting required for a working application, and they can be deployed with no changes. You can

also add new fields and logic to the templates. The templates also extend basic application development capabilities to power users as well.

Domino supports reuse of code that you have created in different ways. Subforms provide a way to share common elements among different forms with the side-effect of enforcing a common look and feel of the application's GUI. Domino applications also can be converted into templates for reuse or adaptation in other solutions.

Domino provides a variety of development alternatives for creating applications, thus enabling you to choose the most appropriate product. The Lotus Notes Designer client is the native development environment for Domino using LotusScript and LotusScript extensions (LSXs) as the development language. However, Lotus has also developed a set of APIs for C, C++, and Visual Basic. Third-party vendors (PowerBuilder, Gupta SQLWindows) have worked together with Lotus to integrate their own development tools.

#### 4.1.3.2 Java Support

Java is a programming language with classes, multithreading constructs, a compiler, and development and debugging tools. You can create Java programs running either as Java applets in a Web browser context or as stand-alone programs (Java applications).

Lotus Notes Designer for Domino introduces several major enhancements related to Java:

- Add Java applets to a Domino application

  Lotus Notes Designer allows Web developers to create a richer, more interactive Web browser and Lotus Notes client user experience by adding support for Java applets. Java applets are now treated like any other Notes object. They can be stored in the Domino object store, they are easily accessible from the menu, and their properties can be easily modified. By storing Java applets within the Domino object store, developers can take advantage of Domino's replication technology for keeping their Java applets synchronized in multiple locations.

  To prevent hostile applets from modifying or destroying information stored in Notes databases, Java applets cannot directly access the Notes Java classes at this time.

  Lotus Notes Designer is not a Java programming environment; therefore you have to develop your Java code outside the integrated development environment (IDE).

- Run Java applets in Lotus Notes Designer for Domino and Lotus Notes clients

  Lotus Notes Designer for Domino and Lotus Notes clients now contain the Java Virtual Machine. So when you place a Java applet in an application being developed, the applet automatically runs in place. Thus you can view and test the applet while designing the application.

- Develop Domino server agents in Java, using the Java interface to the Domino object services

  With Java server agents, Lotus Notes Designer adds another tool to the developer's toolbox. You can now use Java to develop server agents to use along with the in-the-box simple agents, formula agents and LotusScript agents. These agents can perform a range of tasks from e-mail filtering to knowledge management to automated server administration. The agents can be triggered by server events or run as scheduled tasks. The ability to write agents in Java allows you to get the most out of your Java programming resources while still utilizing the power of the Domino environment for your Web needs.

#### 4.1.3.3 JavaScript

Netscape's JavaScript is useful for adding some client-side logic to the Web client. The advantage of using JavaScript is mainly in the improved dynamic behavior of Web pages. JavaScript is not a cross-platform language, and even though other Web browsers, such as Microsoft Internet Explorer, support JavaScript, certain constructs may not be supported.

JavaScript should be used to reduce network traffic between the Web browser and the Domino server. This is not only a performance but also a functional issue. Well-written JavaScript can provide extremely user-friendly context-sensitive help and enable the cursor to be placed in the field that is in error. JavaScript provides some local processing capabilities, such as validations or simple calculations, in the Web browser.

You may also want to use JavaScript when you want to simulate Lotus Notes functional features on a Web browser (for example, dialog and info boxes) to provide a consistent look-and-feel for applications that support both the Notes client and the Web client.

#### 4.1.3.4 Web Development

Domino now more easily handles HTML, allowing you to write or use existing HTML by:

- Adding HTML and typical Web objects to a Web page

You can add HTML to a page by typing it in or by copying existing code into the page and setting it to be *Pass-thru HTML*. Thus it is easy to mix HTML coding with other objects in the Lotus Notes Designer editor.

- *Webifying* Notes objects

  Lotus Notes Designer makes powerful Lotus Notes objects, such as lists of views and navigators, available to Web browsers. Therefore you can embed Lotus Notes objects in Web pages, documents, and forms directly from menu options. In a mixed client environment you can use these Lotus Notes objects to support both Lotus Notes and Web clients.

- Using computed text

  Using computed text, you can present dynamically generated information to your users each time they read a page. Computed text can be mixed with static text and other information on the page. For example, you can compute the current date and time, ask for the user's login name, and even perform lookups in relational databases and present the current value in the Web page.

#### 4.1.3.5 Multiple Clients

Using Lotus Notes Designer, you can develop a single application optimized for both Lotus Notes clients and Web clients, using a single design environment, in a single design session. The result is a single application file or database.

Using the *Hide When* option, you can selectively hide application pieces from either Web browsers or Lotus Notes users. For example, you can use the *Hide When* option in conjunction with HTML code, so that the code will be executed when the application is run with a Web client, but not displayed in a Lotus Notes client.

*On Database Open Launch* options segregate functionality so users can launch a specific application for the Web when run from the Web and another application when run from a Lotus Notes client.

Furthermore application pieces can be hidden either by design or within computed logic on-the-fly.

During development you can prototype Web pages on-the-fly and see what the page will look like within the end user browser environment. Lotus Notes Designer runs an HTTP server locally, launches the Web browser of choice, and displays the current version of the application within the browser.

#### 4.1.3.6 National Languages

If you need to deploy your migrated application in more than one language, you can use Lotus Notes Global Designer to support users worldwide in their native languages. Lotus Notes Global Designer offers you the following key benefits:

- Facilitates customization of applications for specific countries or languages
- Minimizes localization costs and time frames by separating the development and translation processes
- Lets you deploy your Domino application simultaneously worldwide in users' native languages
- Gives multilingual users a choice of the language in which they want to work
- Enables maintenance of multiple language versions of applications with virtually no additional overhead

## 4.2 Lotus Connectors

The Lotus connectors, previously packaged as Domino.Connect, enable Lotus Notes clients and Web browsers to access enterprise data and applications, including enterprise resource planning application systems such as SAP R/3; traditional transaction processing systems such as CICS, MQSeries, and IMS; and relational database systems including DB2, Oracle, Informix, Sybase, and open database connectivity (ODBC) data sources. Figure 5 provides an overview of the Lotus connectors.

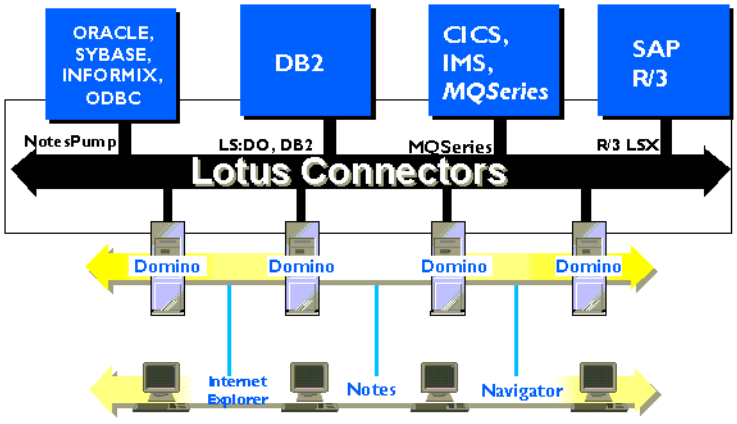



*Figure 5. Lotus Connectors*

### 4.2.1 Why Integration?

Many enterprise applications are based on database technology and transaction processing systems that represent a large investment and offer security, integrity, high availability, and backup and recovery functions. Enterprise applications are often mission critical, and companies rely on them to perform their daily business.

Today companies face the challenge of combining their vast resources of corporate information, and thousands of business transactions within the reach of the Web. A radical modification of enterprise applications is often impractical, yet the risks and costs associated with moving to a new technology base for such applications can be very high.

Frequent and reliable access to corporate data has many benefits, especially accurate, timely data for the users who require it, and the ability to respond to and manage business operations using up-to-date corporate business system information.

Domino and the Lotus connectors enable companies to unlock huge amounts of corporate information and business transactions for a wide audience without having to change the core functions of their enterprise applications.

### 4.2.2 Database System Connectors

With the database system connectors you can replicate data between Lotus Notes and relational databases or query and update data in those databases.

***@DbFunctions***

@DbFunctions can be used for low-volume data inquiries within Lotus Notes or ODBC-compliant database management systems (DBMSs). However, as LSX can perform the same role and provides more flexibility (for example, it can cache the results set and retrieve multiple items), in our migration considerations we do not consider the use of @DbFunctions for looking up DBMS tables.

***LotusScript Data Object***

The LotusScript Data Object (LS:DO) provides full read and write access to external ODBC data sources, using the complete control and flexibility of a structured programming language.

The LS:DO is an object-oriented ODBC implementation that extends LotusScript by providing three classes:

- ODBCConnection

- ODBCQuery
- ODBCResultSet

***DB2 LotusScript Extension***
Based on LS:DO sources, DB2 LotusScript Extension (DB2LSX) offers functions beyond ODBC through the DB2 call level interface (CLI).

DB2LSX is a set of three classes:

- DB2Connection
- DB2Query
- DB2ResultSet

These classes access DB2 data natively. They enable you to use DB2 extended functions, such as support for DB2 user-defined data types and DB2 large objects. They also support the Transact method, which enables you to manage unit of works.

### 4.2.3 Transactional System Connectors

Transaction systems run mission-critical applications and access data within the context of the business rules in today's high-volume production and legacy systems. With the transactional system connectors, you can integrate Domino applications with more than 20 different platforms and systems, including CICS, IMS, HP-UX, Tandem, and Digital.

***MQSeries link LotusScript Extension***
The MQSeries link LotusScript Extension (MQLSX) enables your Domino applications written in LotusScript to communicate, using MQSeries, with applications running in non-Notes environments.

The MQLSX is an API that you call from LotusScript to access the message queue interface (MQI).

The MQLSX provides classes, events, and methods for MQSeries for use from within a LotusScript program.

The MQLSX is designed to:

- Provide an infrastructure to enable you to develop applications that integrate your Lotus Notes environment with your traditional transaction system applications and their data.
- Give you access to all functions and features of the MQSeries API, permitting full interconnectivity with other MQSeries platforms.

- Conform to the normal conventions expected of a LotusScript Extension.

***MQSeries Enterprise Integrator***

The MQSeries Enterprise Integrator (MQEI) extends Domino to include straightforward access to enterprise applications, in a manner consistent with its application development environment.

Features of the MQEI include:

- A set of LotusScript classes. These classes provide a common set of verbs, hiding details of any transaction management the enterprise application is using.
- Built-in capability to access, using MQSeries or CICS, unmodified enterprise applications running under CICS and IMS, as well as native MQSeries applications on other enterprise server platforms.
- Enhanced message-building facilities. Domino programmers can build and interpret messages by reading and writing named fields within the message. The system provides default values for fields that the programmer does not set explicitly, if this is appropriate.
- Integration of the identification and authentication services of Lotus Notes with the access controls provided by CICS and IMS.

The programming model is simpler than the model provided by the MQLSX because the MQEI uses a Lotus Notes database to hold object-definition information.

### 4.2.4 NotesPump

Lotus NotesPump is an enterprise-scale data distribution server that allows you to build applications that read and write data between relational, legacy, and Internet-based data sources.

With the NotesPump server technology, data can be moved or synchronized on a scheduled, one-time, or event-driven basis, or on the basis of a particular business condition. Data can also can be made available to end-user clients on demand, when required, directly from enterprise databases.

NotesPump supports database connectivity among a multitude of major databases, providing native server link connections for Domino, DB2, Oracle 7/8, Sybase System 10/11, Text/ASCII. Two other link types, an ODBC link connection and an EDA/SQL connection, provide NotesPump server access to more than 70 other legacy and sequential file sources.

Another important feature of NotesPump is that, while the server is administered by a Lotus Notes application, NotesPump supports data transfers and replications among dissimilar, non-Notes databases such as from DB2 and Oracle. Therefore, system managers may adopt a single enterprise NotesPump system infrastructure to manage interaction among an increasing number of legacy and relational data sources.

The NotesPump system remains unique in its ability to provide data transfer operations among a growing number of legacy and relational systems without necessarily having to program. Common reusable programming logic is built into the NotesPump product. You can utilize this logic by defining parameters rather than writing raw programming code.

# Chapter 5. Designing a Network Computing Application

In this chapter we discuss several issues to consider when migrating an existing client/server application to a Domino-based network computing application.

There is no right decision for each topic discussed in this chapter. Our objective is simply to provide information to enable you to make design decisions based on your specific conditions.

## 5.1 From CS92 to DOM98: Application Selection

Any application that has distributed processing requirements is a good candidate for development using the network computing model. However, to decide whether to use Domino and its enterprise connectors, you have to determine whether your application:

- Requires accessibility for a large audience through the Internet
- Uses enterprise application services as well as communication, collaboration, coordination, or workflow services
- Requires integration with other Domino databases and applications
- Supports disconnected and/or mobile users
- Supports internal and external users, with different levels of security and role-based access to information
- Supports a variety of clients.
- Handles semistructured or unstructured data in addition to highly structured data
- Supports different languages

CS92 had all of the above characteristics and was a good case study for demonstrating how to migrate to the new network computing model.

As CS92 was developed five years ago, the front-end part of the application had to be redeveloped as there was no longer support or a development environment for it. This situation may often occur in the real world, when choices made in the past have to be reconsidered because of an important change in technology. For implementing our migrated application, Lotus provided us with an IDE that:

- Enabled rapid and effective development
- Offered a variety of development alternatives

- Reduced system management efforts for application deployment

Our migrated application, which we called DOM98 for Domino 1998, was on its way.

## 5.2 DOM98 Infrastructure

In a migration project, the task of defining the system configuration is still necessary. It can simply confirm that the existing configuration satisfies the requirements or, more radically, can lead to reconfiguration to simplify or enhance the operation. The decision is specific to each project, depending on the resources available and the solution chosen.

Figure 6 shows the infrastructure we chose for migrating the application.

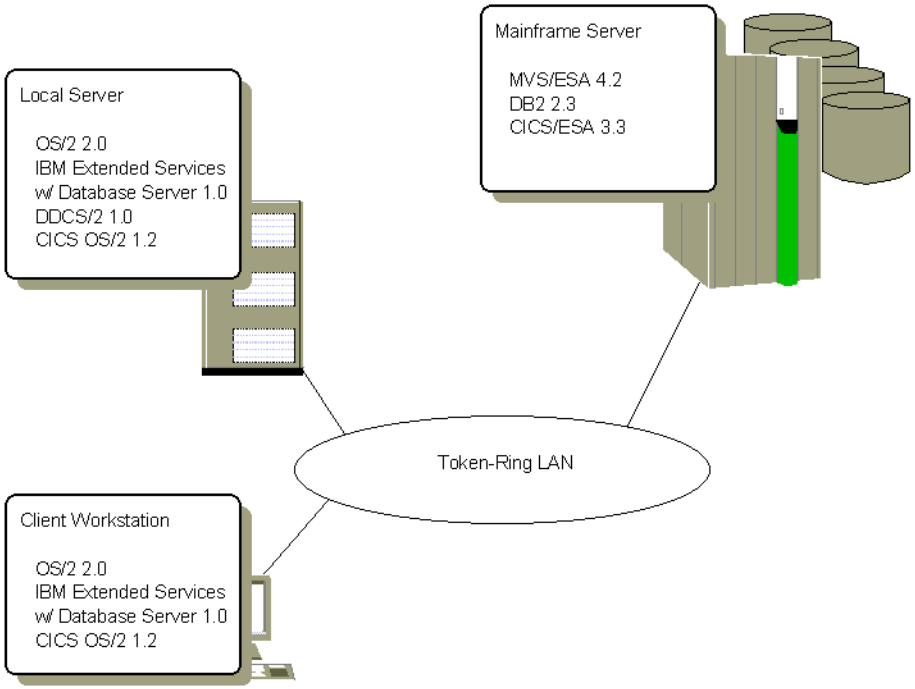



*Figure 6. DOM98 Infrastructure*

All enterprise services, transactions, and databases are now handled by a unique enterprise server using CICS and DB2. We replaced the local server with a Domino server.

We developed a new front end to the application, using Lotus Notes IDE, adding support for mobile users.

To connect to the existing DB2 data and CICS transactions, we used two Lotus connectors:

- MQEI
- NotesPump

Users connect to DOM98 through Lotus Notes clients or Web browsers.

### 5.2.1 Hardware

When looking at the hardware configuration, we had three options:

- A two-tier configuration, where the application using either a Lotus Notes client or a Web browser connects to a mainframe S/390 running the Domino server and the subsystems (DB2 and CICS)

  This configuration has advantages for administration, maintenance, and system management because of the unique server. A unique team could manage that machine. There is only one version of the software, system, or applications. Users have only one entry point to the systems wherever they are located, on the intranet or the Internet. This configuration simplifies the management tasks for CICS and DB2 resources. As all the data and transactions are in the same server, there is no need to support distributed units of works or to create routing functions.

  This configuration, however, has two drawbacks. All of the distributed programs and data have to be relocated to one server, thus leading to many changes in the application programs. As the Lotus connectors are not yet available on S/390, the bridges between Domino and CICS and DB2 have to be developed manually.

- A multitier configuration, where the application connects three servers: the branch office local server, the bank mainframe server, and the branch office Domino server

  This configuration is similar to the client/server configuration, keeping the same distributed structure for data and transactions in two servers, but it is more complex as there is a new Domino server.

  This configuration requires important administrative, maintenance, and system management resources as multiple environments are involved.

- A three-tier configuration, where the application connects two servers: the branch office Domino server for the Lotus Notes application and the enterprise server for the transactions and databases

If the server is on an Intel or UNIX platform, the Lotus connectors are available. In this configuration, you can decide what hardware runs the enterprise services from S/390 to Intel or UNIX servers according to the load. As with the two-tier configuration, administration, maintenance, and system management of CICS and DB2 resources are easier.

In our project, we decided to use the three-tier configuration running the Domino server on an Intel platform.

### 5.2.2 Software

To meet one of the project objectives, we introduced Domino as a major component of the infrastructure. To connect to the existing databases and transactions, we used two Lotus connectors:

- MQEI
- NotesPump

### 5.2.3 Communication

The Internet uses TCP/IP as the communication protocol. We wanted to avoid multiple communication stacks in our environment. Domino and its clients can use TCP/IP to communicate, as well as the client/server communications for CICS and DB2, on any platforms. Therefore, we changed the communication configuration from LU 6.2 and NetBIOS to TCP/IP as shown in Figure 7 on page 63.

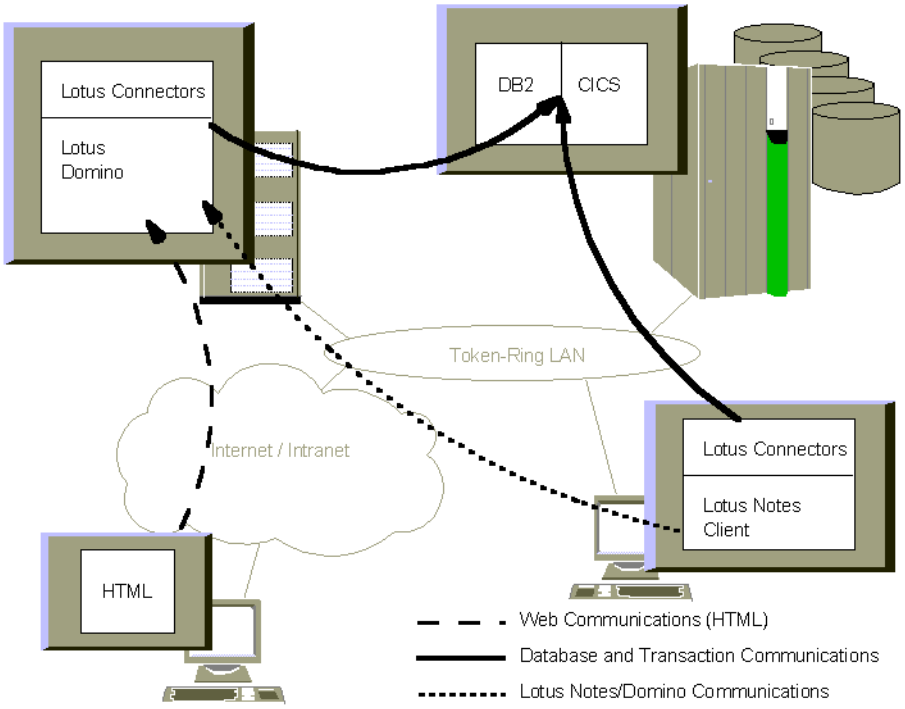



*Figure 7. DOM98 Communication Configuration*

In this environment, all communications—HTML, databases and transactions, and Lotus Notes/Domino—use TCP/IP over the Internet and the intranet. Compared to the CS92 communication configuration (see Figure 2 on page 17), this new configuration is much simpler to manage (one communication stack). It also requires fewer resources (memory and CPU) than the CS92 configuration.

## 5.3 Designing the Architecture

When migrating a client/server application to a Domino-based network computing application, you have to decide whether to migrate the entire application to a Domino application, using a full Domino architecture, or reuse existing parts of the application in an integrated architecture.

### 5.3.1 Full Domino Architecture

In a full Domino architecture, all components are migrated to Domino. The front-end part uses documents and forms, and the business logic is

completely rewritten using LotusScript. This approach is possible only for applications that:

- Require communication, collaboration, or coordination support
- Have few dependencies on other business applications
- Do not need transaction processing and database management support

A full Domino architecture enables you to:

- Exploit Domino services for all parts of the application
- Simplify the system management tasks, as Domino is the unique system
- Reduce potential points of failure, thereby increasing reliability
- React easily to business changes, as fewer components are involved
- Reduce the complexity of accessing underlying business information

Of course, we excluded this option as it would have forced us to rewrite the entire application.

### 5.3.2 Integrated Architecture

In an integrated architecture, the front-end part of the application is migrated to a Domino environment, and the business transactions are split between Domino and the enterprise systems. The Lotus connectors are used to connect the application to the existing enterprise system transactions.

This approach allows you to gain such advantages of a Domino application as communication, collaboration, and coordination support as well as mobile or disconnected user support.

The integrated architecture approach also allows you to gain such advantages of the enterprise OLTP and DBMS as:

- High transaction rates
- Large volumes of structured data
- Concurrent updates
- Backup and recovery

The business transactions may need to be divided into two subtransactions, one running in the Domino environment and the other running in the enterprise environment. For example, an order transaction may be divided between:

- A reservation step, occurring immediately in the Lotus environment

- A processing step, occurring later in the enterprise environment

This transaction split allows you to support mobile or disconnected users as the integration with the enterprise application is asynchronous (see “Asynchronous Integration” on page 68).

## 5.4 Designing the Enterprise Connection

To connect to the enterprise, an application can use either a direct connection from the Lotus Notes client workstation or a gateway connection through the Domino server.

### 5.4.1 Direct Connection

A direct connection is set up between the Lotus Notes client and the enterprise system (see Figure 8).

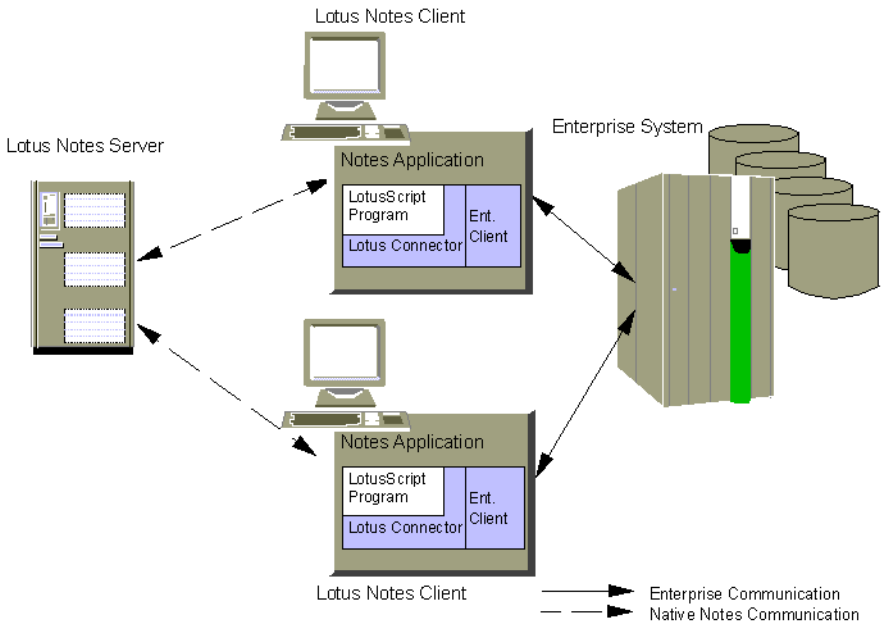



*Figure 8. Direct Connection*

On a direct connection, the sequence of actions is:

1. The user enters a request for the enterprise system. (The term *request* is used here for any action on the enterprise system.)

2. The LotusScript code running on the Lotus Notes client uses the Lotus connector classes to build a connection to an enterprise system.
3. The Lotus connector classes convert the entered request data such that it is usable by the enterprise system. The request is sent to the enterprise system.
4. The LotusScript code uses the Lotus connector classes to receive the reply from the enterprise system and to convert the reply data such that it is usable by Lotus Notes. The reply could be data that is queried, a return code, or any other reply from the enterprise system.
5. The reply data is displayed to the user.

To set up a direct connection, you need to install the enterprise client— such as the CICS client, the DB2 client, or the MQSeries client— and the Lotus connector classes on each Lotus Notes client workstation that establishes a connection to the enterprise service. Apart from the fees for the licences, this management may be complex as you have to:

- Install and configure the connection software and the Lotus connector on each Lotus Notes client.
- Establish two connections for each client, one connection to the Domino server and the other to the enterprise application server.
- Apply changes on each client, especially if they are outside your Lotus Notes. (In this case, using templates or replication does not help.)

### 5.4.2 Gateway Connection

A gateway connection is set up between the Domino server and the enterprise service. The Lotus Notes client or the Web browser triggers a server agent (see Figure 9 on page 67).

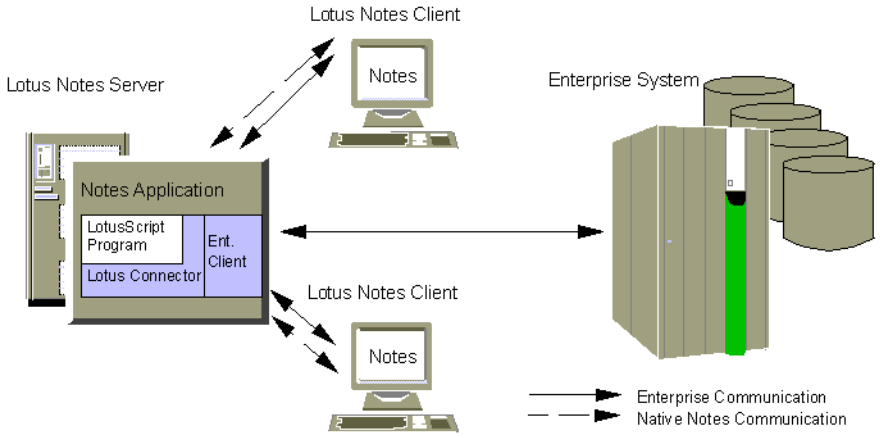



*Figure 9. Gateway Connection*

In a gateway connection, the sequence of actions is the same as that of the direct connection, supplemented by the two steps highlighted below:

1. The user enters a request for the enterprise system.
2. **The request data is transferred from the Lotus Notes client to the Domino server.**
3. The LotusScript code running on the Domino server uses the Lotus connector classes to build a connection to an enterprise system.
4. The Lotus connector classes convert the entered request data such that it is usable by the enterprise system. The request is sent to the enterprise system.
5. The LotusScript code uses the Lotus connector classes to receive the reply from the enterprise system and to convert the reply data such that it is usable by Lotus Notes. The reply could be data that is queried, a return code, or any other reply from the enterprise system.
6. **The reply data is transferred from the Domino server to the Lotus Notes client.**
7. The reply data is displayed to the user.

A gateway connection requires neither the installation of the enterprise client nor the installation of the Lotus connector classes on each Lotus Notes client workstation. Instead they have to be installed only on the Domino server where the LotusScript code runs that uses the Lotus connector classes to establish connections to the enterprise service. The system management

tasks are simplified as you have to manage only the installation and configuration of the Domino server and its connectors.

## 5.5 Designing the Integration

There are two ways to integrate Domino and the enterprise parts of the application:

- Asynchronously, if the application supports disconnected users who may not wait for the answer from the enterprise system
- Synchronously, if the application requires online processing

### 5.5.1 Asynchronous Integration

Use asynchronous integration if your application does not require immediate responses from the enterprise application. In this model, the Lotus application issues a request for the enterprise system and returns to its own process. It does not have to wait for a response from the enterprise application. Later on when the enterprise application has completed, both partners have to resynchronize. The asynchronous integration is similar to an enterprise application's online transaction program starting a batch job.

Asynchronous integration and a gateway connection are often implemented together. But asynchronous integration might also be considered useful as a backup solution for synchronous integration, if the enterprise application service or the network connection is temporarily down.

### 5.5.2 Synchronous Integration

Use synchronous integration if your application requires an immediate response to a request that has been sent to the enterprise application. In this model, the Domino application issues a request for the enterprise system and waits for the response from the enterprise application.

With a Lotus Notes client, synchronous integration can be achieved only if the code handling the request to the enterprise application runs locally on the client that is using a direct connection. Triggered by a Lotus Notes client, a server agent on the gateway would be started with some delay.[1]

With a Web browser, you can use a gateway connection to achieve synchronous integration because the agent triggered by the Web browser on the Domino server is started immediately.

[1] Domino Release 4.6.1 includes a new RunOnServer method of the Domino server's agent class. Using this new method, you can start an agent without any delay on the server.

## 5.6 Data and Function Placement

Location transparency is still a fundamental requirement for network computing applications. Flexibility is needed, and it is important to be able to change the location of data or function without changing the code. The Domino replication feature supports location transparency as data and application function can be replicated to different Domino servers and therefore be close to the user.

### 5.6.1 Data Flexibility

Full transparency of data is not always easily attainable. In the case of CS92, for example, a major part of the DB2 table design was to determine where the data should be placed. As a result, tables that were specific to a geographic location, such as a bank branch, did not carry a branch identifier. If this sort of table is migrated from the server to the mainframe, an extra column must be added to the table, and application code must be changed accordingly.

### 5.6.2 Function Flexibility

Transparency of application function is achievable by physically moving the code from one platform to another, or by using system software such as CICS or DB2 to perform remote access. The location of code should not be volatile, but requirements such as performance tuning and load balancing may necessitate the moving of code; hence, a means of routing client requests to the appropriate functions must be provided.

### 5.6.3 CS92 Application Request Manager

CS92 used the application request manager (ARM) to support the flexibility requirement. The ARM stands between the application logic on the client workstation and the distributed system services such as access to DB2 data or a call to a CICS transaction. The ARM has two functions:

- It verifies that the application logic is requesting a known service and using the correct parameter for that service.
- It routes the request through the system service interface that handles the particular service requested.

The ARM was implemented in CS92 for two reasons:

- It served as the interface between the EASEL and COBOL code.
- It protected the client from needing to know the location of the server code.

We did not need to use the ARM as the interface in our migration because LotusScript makes the equivalent calls through the Lotus connectors. However, client protection was more relevant to our migration. For example, the client requests the GetCashierID service, so the ARM determines which program it requires and which service provides it. The developers recognized that directory services external to the ARM should be used to hold the precise location information, but time constraints prevented them from implementing this.

### 5.6.4 Data Placement

Data placement is a simple task in a network computing application as there is only one place—the DBMS—to store the operational data, unlike the three-tier model of the client/server application. We also considered that all operational data had to be managed by the DBMS. However, as some data might be duplicated from DB2 to the Domino database, and further to the Lotus Notes client, we considered the DB2 data as master in case of a conflict or a restoration.

In CS92, data located on the workstation was specific to the user. In DOM98, we moved all data stored previously on the client to the enterprise server machine. To migrate these tables, we added an additional column to the table so that specific instances of a currency could be identified with a specific cashier. Therefore, all the tables stored on the client machine must be listed, and data must be analyzed to determine which new key information is to be added to a table to uniquely identify a row.

In a client/server environment, it is customary to hold reference data on the client machine for ease and speed of access. The data is usually maintained on the server machine and downloaded to the client. The frequency of this downloading is dependent on the volatility of the data. In DOM98, to support the reference data, we used local copies of data. Using NotesPump, we periodically replicated DB2 tables to the Domino server. Then we replicated Domino databases from the server to the clients, using the Lotus replication feature.

Moving the data to the enterprise server forced us to change the tables, and thus the COBOL code, adding the cashier and branch IDs as part of the unique key.

Moving the reference data from DB2 to Lotus Notes implied simplification in the application, replacing some CICS transactions with LotusScript code to access the local data. Using Lotus Notes reference data allowed us to support mobile and disconnected users.

### 5.6.5 Function Placement

CS92 processed transactions on the branch office and mainframe servers. It used the ARM to support the flexibility requirement.

DOM98, using a unique database and transaction server, does not require such a function. However, we could have implemented it very easily with our Lotus connectors:

- MQEI uses the definition database to locate the services the application accesses.
- NotesPump defines a link for each DBMS to which it connects.

In DOM98, we had the choice of keeping the existing transactions and deciding on their placement or migrating the functions to a LotusScript code and placing them within the Lotus Notes database. We identified the application functions that could be reused with little or no changes and decided to use them as is on the enterprise server.

Some functions had to be rewritten because:

- The programming tools used previously were no longer available (the CS92 front end was written in EASEL, which is no longer supported).
- The requested skills were no longer available.
- Application enhancements such as e-mail and cooperation were required.

We used LotusScript to deliver enhanced groupware functions and the Lotus connectors to bridge to the existing enterprise application.

Function placement also depends on whether the function implements:

- An input validation and simple calculation

  Functions checking user input or performing simple calculations should reside on the client because this placement results in reduced network traffic.

- A bridge to the enterprise application

  Placement of functions that implement a bridge to the enterprise application depend on the connection model that your application is using. With a direct connection, these functions reside on the client, whereas with a gateway connection they reside on the Lotus Notes server.

- A Lotus Notes enhancement

  Lotus Notes functions such as mobile user or workflow support should reside on the Domino server.

- A data update

  Functions updating data should reside on the server that manages that data. All applications that need to update that data will use that unique function, improving the integrity of the data and facilitating its management.

## 5.7 Designing for Users

A Domino application supports the following clients:

- Lotus Notes clients, using either a direct or gateway connection
- Web browsers, using a gateway connection

From an application point of view, a Domino application reaches the following users:

- Stationary users, who have a permanent connection to the Domino server. They can use either a Lotus Notes client or a Web browser to access the application. For example, professionals working in an office or customers and suppliers using the Internet are typical stationary users.
- Mobile users, who are occasionally connected to the Domino server. Typical mobile users travel around, working at home or a customer's site. They can only use a Lotus Notes client. They establish connections to their Domino server from time to time to synchronize their local database replicas with the server databases.

### 5.7.1 Stationary Users

Stationary users can use either a Lotus Notes client or a Web browser to access the Domino application.

The Lotus Notes client supports synchronous integration if the code needed to connect the enterprise system is running locally. With a Lotus Notes client, the application:

- Integrates communication, collaboration, coordination, and complex workflow
- Supports Lotus Notes replication
- Uses tailored views to customize the available information, thus supporting its specialized tasks

The Lotus Notes client requires a session with the Domino server.

The Web browser might be the lowest cost alternative in terms of installation, operating, and maintenance costs. It supports synchronous and asynchronous connections through a gateway connection. With a Web browser, the application supports external user access to the application. However, to implement workflow, the Web users must be identified.

There is no way to keep a permanent session between a Web browser and the Domino server. So, in a multistep operation, you must save intermediate documents to keep track of the previous step. To delete obsolete documents left by users leaving the application, you may have to develop a garbage collection agent.

You should also consider the specific development issues for the Web Client, such as the use of the Notes user interface classes and Notes message boxes. *Developing Web Applications Using Lotus Notes Designer for Domino 4.6* covers in detail the development of applications for Web clients.

### 5.7.2 Mobile User

Mobile users use a portable workstation with a full Lotus Notes client environment. When not connected, the application stores the user's request in the local database. At connection time, the database is replicated on the Domino server, and the processing request can be transmitted to the enterprise service.

The answers, to the enterprise requests are transmitted back to the mobile users at the next replication between the Domino database and the local database.

Support for mobile users requires an application that uses:

- An asynchronous integration model where immediate replies are not mandatory, such as replies are not possible when a connection is not available
- A user action, an agent, or a replication function on the server to process the requests that have been replicated from the user's database
- An agent or function that presents to the user the results of the requests
- An error handling method that supports the connection errors

Mobile users also have to be able to work on data that is not always up-to-date, so the business should tolerate minor data inconsistencies. For example, it may be acceptable to work on a product set that does not contain all product descriptions, whereas it is unacceptable to work on stock out-of-date information.

Domino applications supporting mobile users can use the direct or gateway connection. The main difference is the type of network connection that is required:

- A direct connection requires a remote LAN service connection

  A remote LAN service connection uses the protocol that the network uses, thus enabling you to establish a connection with your enterprise application, using the workstation's back-end client software. Additional software is required to establish the remote LAN connection.

- A gateway connection can use a dial-up modem connection

  Using modems and the Lotus Notes XPC protocol, this connection lets you access your Domino server through a modem when your workstation is not connected to a LAN. This is the most common remote connection for mobile Lotus Notes clients because it does not require any software other than Lotus Notes.

## 5.8 Designing the User Interface

To users of a system, the migration from a client/server application to a network computing application should not detract from their ability to perform the task at hand. However, a migration to network computing is a good time to reevaluate the GUI. As the Internet opens new channels into your business processes, you cannot make the same assumptions about its user base as an Intranet-based GUI. For an intranet-based application, the GUI is a tool to complete the business process in the most efficient manner. For an Internet-based application, in addition to the usability aspect of the application, you must be aware that anyone using your GUI will be forming opinions about the company.

In addition to these considerations, Lotus Notes Designer provides features that facilitate the use of the applications:

- Navigators provide a graphical way to find documents or take actions without having to maneuver through views or find the commands on the pull-down menus.
- Views present summary information in rows and columns so that you can find the documents you want to read.
- Forms present the data of a document or create a document and enter data.
- Fields, especially the following types:

- RichText fields to enrich the existing information by adding graphics, video, or audio objects
- DocLinks to store a reference to some related information, thus implementing an effective way of navigation
- Collapsible sections to improve a document's structure

## 5.9 Designing for Security

Security is a critical issue in any application as only authorized users should be able to access applications and data.

Client/server applications implement security on three levels:

- The transaction level, which secures the application programs used in enterprise service transactions
- The resource level, which secures the resources that the application programs are going to use (for example, DB2 resources)
- The connection level, which secures the communication link between the programs.

### 5.9.1 Domino Implementation

Domino offers different security level options, depending on whether the access is from the Lotus Notes client or the Web browser. This security setup also effects the type of connection.

#### 5.9.1.1 Server Security

In addition to authenticating users and checking whether they are authorized to access the resources they are asking for, Domino provides the following security levels:

- Database level security, which uses the database's access control list (ACL) to assign different access levels to different users or groups of users

  You can use this feature to provide editor access to the internal users and read access to external users, preventing them from changing the contents.

- Form and view level security, which restricts the use of a particular form or view

  With this security level, you can manage the distribution of sensitive information in an intranet.

- Document level security, which allows the owner of a document to restrict access to specific users or groups of users, in both read-only and update mode.

  You can use this level to authorize only the creator of a document to modify the document.

- Field level security, which restricts access to certain fields of documents to specific users or groups of users, in both read-only and update mode.

  You can use this level to allow different groups to update different fields in the same document. For example, in an employee document, the salary field can be updated only by the finance department staff, whereas the employee information fields can be changed only by human resources staff.

#### 5.9.1.2 Lotus Notes Client

The Domino server authenticates Lotus Notes client users through certificates based on public and private keys. It compares the certificate presented by the user with certificates in its possession. If any of the certificates presented by the user were issued and signed by an entity that issued and signed one of the server's own certificates, or if the server issued a certificate to the user, the server can use that fact to establish the identity of the supplicant. This certification scheme is a common method of authentication, used not only by Lotus Notes but also in other software security implementations.

#### 5.9.1.3 Web Client

Unless you want all Web users to access the Domino server anonymously, you must set up Web user authentication at the server. For each Web user who is allowed to access the Domino server, you have to create a Person document and add an HTTP password. The Domino Web site has a sample registration application for you to download.

Although basic Web authentication is not considered as secure as Lotus Notes public key certificate-based authentication, as it does not involve a certifier who validates the user's identity, it does allow Domino to serve users through the Internet.

#### 5.9.1.4 Encryption

Domino offers encryption to secure the transmission of data through the network.

For Lotus Notes clients, encryption is implemented by using a pair of matched keys. Messages are encrypted and decrypted with a public key and a private

key. One key is useless without the other; that is, a message encrypted with a public key can only be decrypted with the associated private key.

For Web clients, encryption is implemented through the SSL protocol, which provides communications privacy and authentication over the Internet. When the Domino Web server is configured for SSL transactions, it can encrypt data that passes between Web clients and the server. Enabling SSL ensures that HTTP data is encrypted to and from clients. Privacy is ensured during transactions as an encoded message digest accompanies data to detect any message tampering, and the server's digital signature accompanies messages to assure clients that they received their messages from the correct server.

### 5.9.2 Connection

There are some security impacts to consider for connections between the Lotus Notes and enterprise system environments:

- Direct connection

  For a direct connection to the enterprise service, the application can use the user ID and password of the requesting user (see Figure 10). For example, to access a DB2 database through DB2LSX or LS:DO, each user must provide a user ID and password at the connection step. If you do not provide one, you are prompted for one.

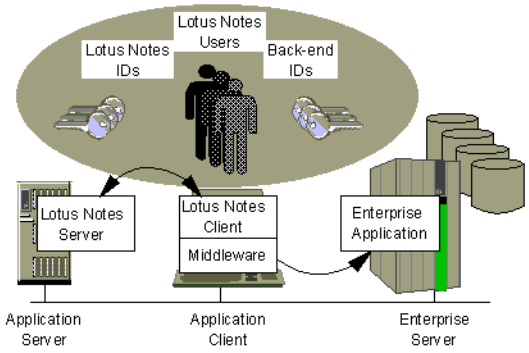



*Figure 10. Direct Connection*

- Gateway connection

  For a gateway connection to the enterprise service, the application uses a user ID and password, generally from the server (see Figure 11 on page 78). This is especially important for Web users as there is no way, other than programming, to prompt them for identification

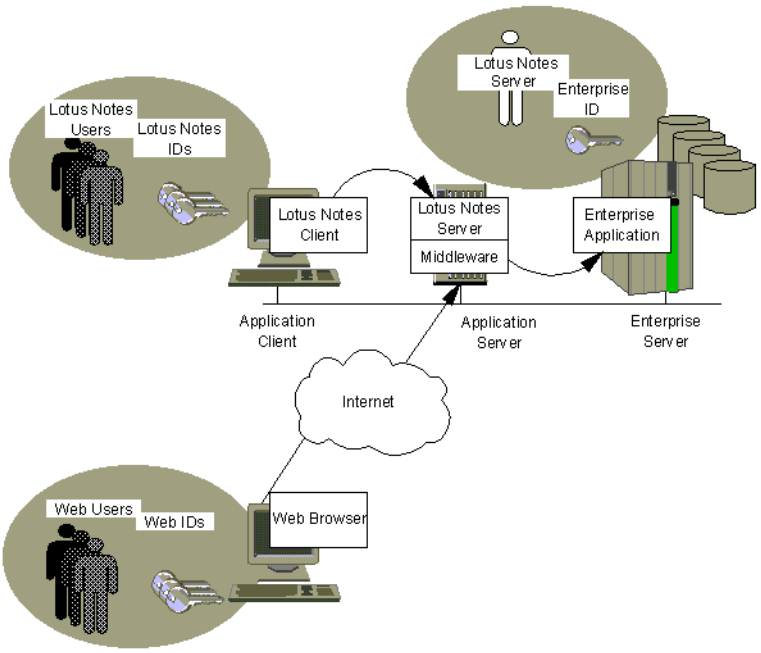



*Figure 11. Gateway Connection*

These two approaches have the following impacts:

- From a user perspective

  The direct connection may be confusing because it requires two IDs to log on to the same application, one for the Lotus Notes environment (this also applies for the Web user if anonymous access is not provided) and one for the enterprise application. As implemented by the MQEI, you can use a database to map the Lotus Notes and enterprise IDs. This prevents prompting users twice for their user ID. It also requires that users update their entry in the mapping database whenever they change the password for their enterprise application's user ID.

  The gateway connection is more simple as only the server's administrator has to maintain the ID that is used for identification at the enterprise application level.

- From a control and audit perspective

  The direct connection offers full logging capability at the enterprise application level because each user can be correctly identified.

Enabling audit with the gateway connection requires additional coding because Lotus Notes does not provide a logging function. Therefore you have the choice of developing additional code that enables logging either at the Domino server level or the enterprise application level.

- From a security perspective

  To implement the gateway connection, you may need to hard code the user ID and password in the LotusScript code. In such a case, anyone who has read access to the LotusScript code is in possession of a key to your enterprise application. This fact might be even more important if your application is accessible not only from the intranet but also from the Internet.

## 5.10 Designing for System Management

When migrating to a network computing environment, the new Domino application and its enterprise connection have system management impacts. Depending on how many client workstations need to be managed, a system administration and software distribution environment must be installed.

In a direct connection, you have to install and configure the appropriate Lotus connector and the enterprise client middleware on each client workstation as in a gateway connection only the server needs the software.

Change management is an important topic, especially if the changes are outside your Domino application and cannot be replicated from a Lotus Notes database. Tracking and analyzing problems is also complex as the application involves multiple environments (for example, Domino, enterprise service, and connectivity.)

## 5.11 Prototyping

Prototype issues are the same in a network computing environment as in a client/server environment. The use of the Lotus Notes integrated development environment enables rapid prototypes of screens and associated connections to the enterprise applications.

Prototyping is useful for presenting the new application to users so that they can agree on its look and feel and ensure that it delivers the requested functions.

Prototyping is also useful for iteratively implementing the application parts and establishing the associated bridges to the enterprise applications. Special care is then given to the redeveloped parts of the application.

In the area of prototyping, Lotus Notes Designer helps you design and debug code by:

- Enabling application previewing in a Lotus Notes client or in a common Web browser

  You can prototype Web pages on-the-fly and, as you are working, see what the page will look like within the end-user browser environment. Lotus Notes Designer runs an HTTP server locally that launches the Web browser of choice and then displays the current version of the application within the browser.

- Providing a Java Virtual Machine

  When you place a Java applet in an application under development, the applet automatically runs in place.Thus you can view and test the applet while designing the application.

# Chapter 6. Designing the New Lotus Application

In this chapter we describe how we used the MQEI, NotesPump, and the Lotus Notes database to develop the DOM98 application.

## 6.1 MQEI

In CS92, the client application used distributed program link (DPL) to call CICS OS/2 programs. The CICS communication area (COMMAREA) was used to exchange the data.

In DOM98, we developed the client application with LotusScript. The client application uses MQEI to connect an enterprise service: a CICS DPL program on the enterprise server through the CICS client using the CICS external call interface (ECI).

### 6.1.1 System Structure

Figure 12 Illustrates the MQEI system structure.

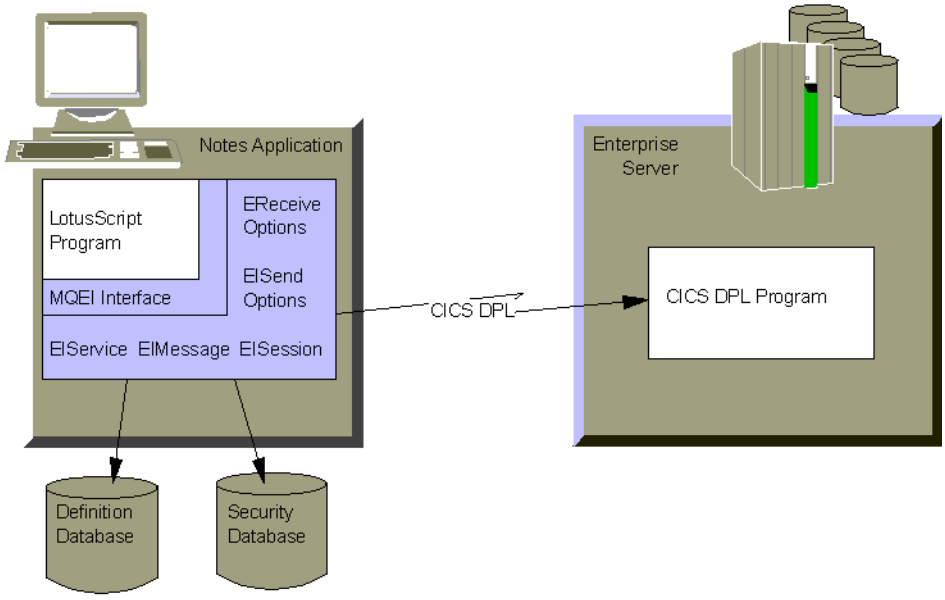



*Figure 12. MQEI System Structure*

The MQEI provides an LSX that gives the LotusScript programmer access to the MQEI interface. The two key classes from this interface are the EIService and EIMessage classes:

- The EIService class provides a universal interface suitable for driving a variety of styles of enterprise services. In DOM98, we used the CICS DPL direct service.

  The service is defined in the MQEI definition database. The definition for an EIService object contains information such as the type of transaction system on which it runs, and the communications mechanism used to access it.

- The EIMessage class encapsulated the parameter (field) flows to and from the enterprise application.

  Messages are also defined in the MQEI definition database. A definition consists of a set of named fields of specific type (for example, string) and mapping information stating how the individual fields are to be mapped into data structures understood by the existing enterprise service.

### 6.1.2 Security Database

Access to enterprise services can be restricted to certain password-protected user IDs defined and managed on the enterprise system. With CICS transactions, these user IDs are managed by the CICS security system (CICS itself, RACF on an S/390, User Profile Manager (UPM) on OS/2). Lotus Notes has its own extensive security mechanism that uses both passwords and public key technology. MQEI brings both together through its MQEI security database, a Notes database containing the host user IDs and authenticators (passwords) an end user will need to run enterprise transactions. As the authenticator itself is sensitive information, the content of the Authenticator and Verify Authenticator fields are encrypted, and Notes security only allows the contents to be entered and modified by that Notes user specified in the entry.

When an MQEI application wants to communicate with an enterprise application, through an EIService object, the MQEI extracts the user ID and password required from the MQEI security database and uses them to authenticate the end user with the enterprise system. This authentication is performed by CICS and is transparent to the MQEI application. In order for the MQEI to extract information from the MQEI security database, each user of the application (and the Domino server for server agents) must be given Author access in the ACL of the MQEI security database.

In the MQEI security database we created a security definition for each user who has access to the enterprise service from a Lotus Notes client (see Figure 13).

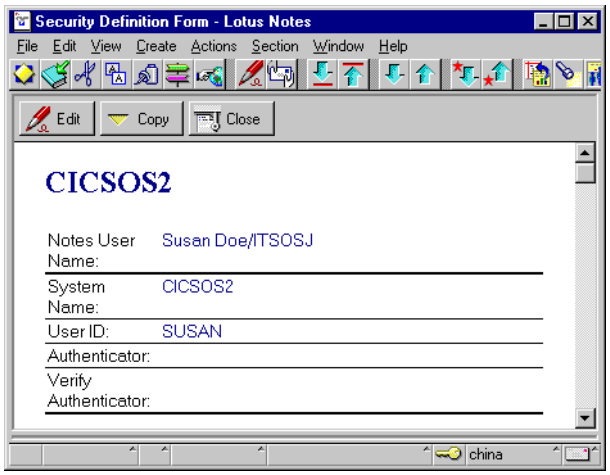



*Figure 13. MQEI Security Definition for a Cashier*

We also created a security definition for the Domino server to support Web users. This definition is necessary for the support of server agents (see Figure 14).

.

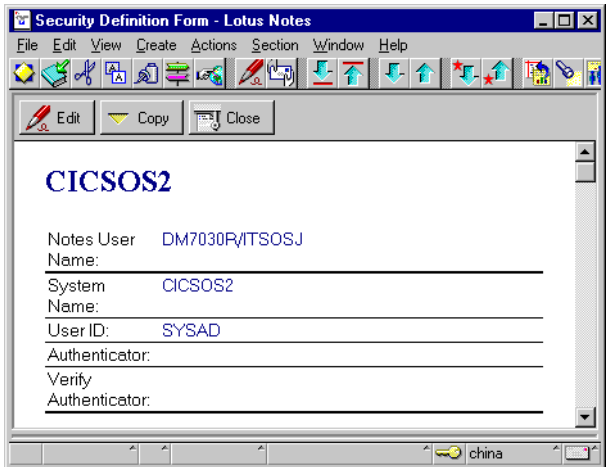



*Figure 14. MQEI Security Definition for the Domino Server*

### 6.1.3 Definition Database

The MQEI can be utilized from simple LotusScript programs. In DOM98, to access the enterprise services, we created LotusScript functions—routines returning a value—that use services and messages defined in the MQEI Definition database.

To connect to the CICS DPL program on the enterprise server, the MQEI uses the ECI. It uses the COMMAREA to send to and receive data from the CICS DPL program.

Depending on the enterprise program, the COMMAREA may be simple or complex as is the corresponding MQEI message. In the sections that follow, we explain how to:

1. Read and write a simple message
2. Read a complex message
3. Write a complex message

Appendix A, “Script Library” on page 115, shows all of our sample LotusScript functions to access MQEI services and messages.

#### 6.1.3.1 Read and Write a Simple Message

The GetCashierID service returns a cashier ID for the selected cashier name. It connects to CICS program ASJB09C, using the simple COMMAREA shown in Figure 15.

```
typedef struct
{
     UCHAR PROGRAM_NAME [8];
     UCHAR MESSAGE_COMMAREA [80];
     UCHAR CODE_COMMAREA [7];
     UCHAR STATE_COMMAREA [5];
     UCHAR NAME [40];
     UCHAR IDENTIFICATION [4]
} CASHIERID;
```

*Figure 15. Simple COMMAREA*

***Service Definition***

Figure 16 on page 85 shows the service definition for GetCashierID.

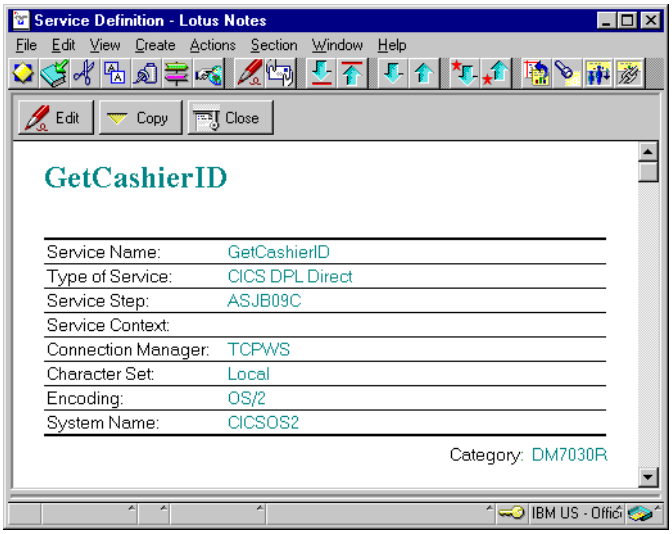



*Figure 16. Service Definition: Simple Message*

- Service Name defines the name of the MQEI service, GetCashierID, the LotusScript application uses.
- Service Step defines the name of the enterprise CICS DPL program that is executed. MQEI service GetCashierID connects to CICS DPL program ASJB09C.
- Connection Manager gives the name of the enterprise CICS server, TCPWS, as defined in the CICSCLI.INI file of the CICS client (see Appendix D, “CICSCLI.INI” on page 135).
- System Name locates an entry in the MQEI security database.

### *Message Definition*

Figure 17 on page 86 shows MsgGetCashierID message definition. This message is used by MQEI service GetCashierID to exchange data with the enterprise CICS program. This message definition corresponds to the CICS COMMAREA.

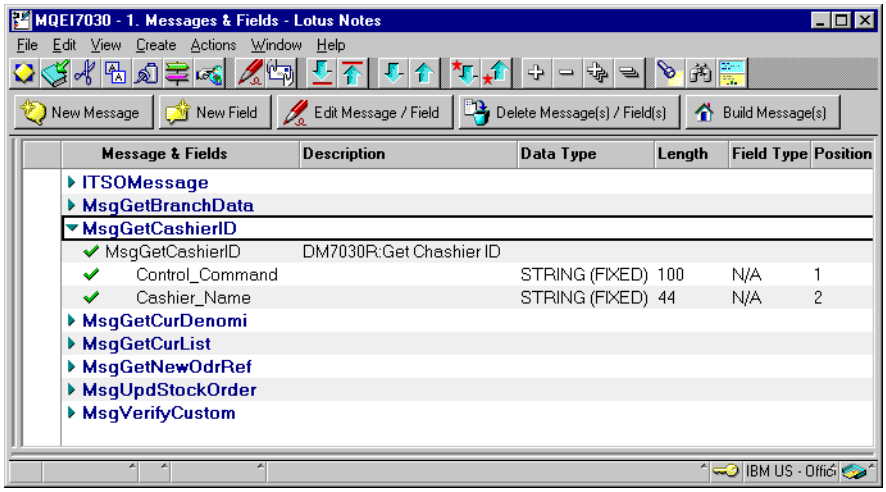



*Figure 17. Message Definition: Simple Message*

In MsgGetCashierID, we create two message fields:

- Control_Command corresponds to the system data set automatically by the MQEI.
- Cashier_Name corresponds to the application data, the cashier name as input and the cashier ID as output (see Figure 18)

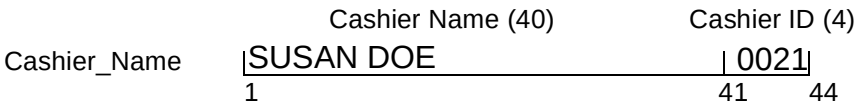



*Figure 18. Cashier_Name Field in MsgGetCashierID*

We merged the fields of this simple COMMAREA into two MQEI fields in order to use the same programming structure as for a complex message.

### *LotusScript Function*

In the LotusScript function, we performed the following steps:

1. We created EISession, EIService, EIMessage, EISendOptions, and EIReceiveOptions objects and set the cashier name in the MQEI message:

```
Set MySession = New EISession
Set MyService = MySession.CreateService("GetCashierID")
Set MyMessage = MySession.CreateMessage("MsgGetCashierID")
Set MySendOptions = MySession.CreateSendOptions
Set MyReceiveOptions = MySession.CreateReceiveOptions
```

```
Mymessage.Cashier_Name = CashierName ’Input Cashier Name
```

2. We connected to the enterprise service, using the MQEI service, sent the message, and waited for a reply:

```
MyReceiveOptions.WaitType = EIWT_WAIT
MyReceiveOptions.WaitInterval = 1000
Call MyService.Connect
Call MyService.SendMessage (MyMessage, MySendOptions)
MyReceiveOptions.Identifier = MySendOptions.Identifier
Do
    Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
    If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
        retcode = False
        Exit Do
    Else
        retcode  = True
        Exit Do
    End If
Loop
```

3. We parsed the reply message to get the cashier ID from the response string, using the Mid$ function:

```
rcvdata = MyMessage.Cashier_Name
CASHIER_ID= Mid$(rcvdata,41,4)
```

#### 6.1.3.2 Read a Complex Message

The GetCurList service returns a list of available currencies. It connects to CICS program CSJB03C, using a complex COMMAREA shown in Figure 19.

```
typedef struct
{
    UCHAR PROGRAM_NAME [8];
    UCHAR MESSAGE_COMMAREA [80];
    UCHAR CODE_COMMAREA [7];
    UCHAR STATE_COMMAREA [5];
    STRUCT
    {
        UCHAR TYPE_OF_CURRECY_C [3];
        UCHAR COUNTRY_C [20];
        UCHAR ACTUAL_STOCK_LEVEL [12];
        UCHAR RECOMMENDED_STOCK_LEVEL [12];
        UCHAR CURRENCY_RESTRICT_INFO [80];
        UCHAR FOREIGN_INFO [80];
    } CURRENCIES [30];
} CURRENCYLIST;
```

*Figure 19. Complex COMMAREA (Read)*

#### Service Definition

Figure 20 on page 88 shows the service definition for GetCurList.

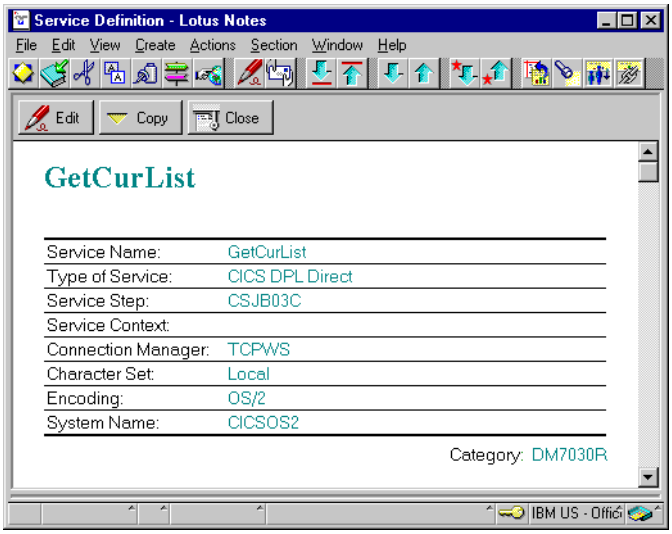



*Figure 20. Service Definition: Complex Message (Read)*

MQEI service GetCurList connects to CICS DPL program CSJB03C.

### *Message Definition*

Figure 21 shows MsgGetCurList message definition. This message is used by MQEI service GetCurList to exchange data with the enterprise CICS program. This message definition corresponds to the CICS COMMAREA.

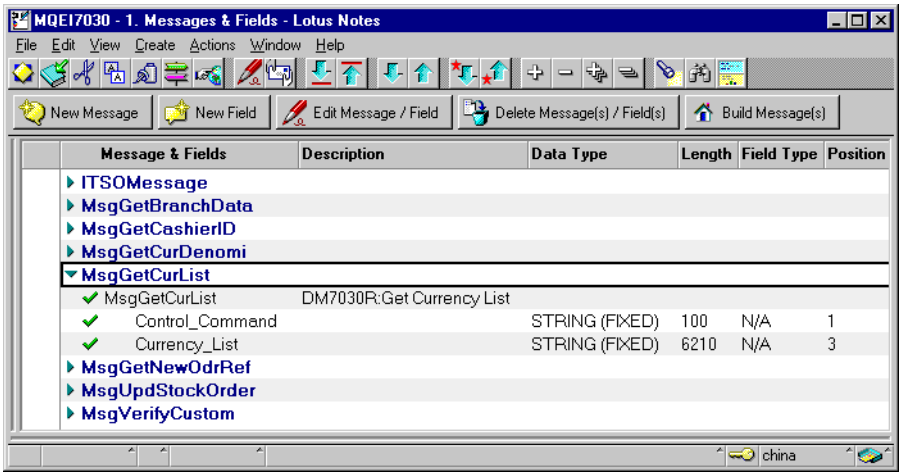



*Figure 21. Message Definition: Complex Message (Read)*

In MsgGetCurList, we created two message fields:

- Control_Command corresponds to the system data set automatically by the MQEI.
- Currency_List corresponds to the data of 30 currencies as output (see Figure 22). The Currency_List length is 6210 bytes (207 x 30).
)

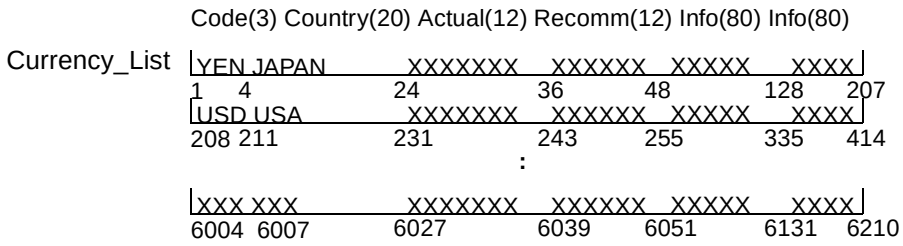



*Figure 22. Currency List Field in MsgGetCurList*

### *LotusScript Function*

In the LotusScript function, we performed the following steps:

1. We created EISession, EIService, EIMessage, EISendOptions, and EIReceiveOptions objects. The message was empty as there was no input for that function:

```
Set MySession = New EISession
Set MyService = MySession.CreateService("GetCurList")
Set MyMessage = MySession.CreateMessage("MsgGetCurList")
Set MySendOptions = MySession.CreateSendOptions
Set MyReceiveOptions = MySession.CreateReceiveOptions
```

2. We connected to the enterprise service, using the MQEI service, sent the message, and waited for a reply:

```
Call MyService.Connect
Call MyService.SendMessage (MyMessage, MySendOptions)
MyReceiveOptions.Identifier = MySendOptions.Identifier
Do
    Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
    If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
        retcode = False
        Exit Do
    Else
        retcode = True
        Exit Do
    End If
Loop
```

3. We parsed the reply message to create a LotusScript array of all currency data from the response string, using the Mid$ function:

```
rcvdata = MyMessage.Currency_List
For count = 0 To 29
    curtable(count) = Mid$(rcvdata, (count)*207 + 1, 207)
Next
```

4. We created LotusScript lists to be used as keywords in the application; each item is separated by a semicolon (;).

```
For count = 1 to 30
    CUR_TYPE = CUR_TYPE + Left$(curtable(count), 3) + " - "
    CUR_TYPE = CUR_TYPE + Trim(Mid$(curtable(count), 4, 20)) +";"
    CUR_RECM = CUR_RECM + Trim(Mid$(curtable(count), 24, 12)) + ";"
    CUR_ACTS = CUR_ACTS + Trim(Mid$(curtable(count), 36, 12)) + ";"
    CUR_INFO = CUR_INFO + Trim(Mid$(curtable(count), 48, 80)) + ";"
    CUR_FORG = CUR_FORG + Trim(Mid$(curtable(count), 128, 80)) + ";"
Next
```

#### 6.1.3.3 Write a Complex Message

The UpdateStockOrder service sends an order of currencies from the cashier. It connects to CICS program CSJB07C, using a complex COMMAREA shown in Figure 23. The item in bold represents the data, the request from the branch office, given by the LotusScript application.

```
typedef struct
{
    UCHAR PROGRAM_NAME [8];
    UCHAR MESSAGE_COMMAREA [80];
    UCHAR CODE_COMMAREA [7];
    UCHAR STATE_COMMAREA [5];

    UCHAR CASHIERID[4]
    STRUCT
    {
        UCHAR TYPE_OF_CURRENCY_D [3];
        UCHAR COUNTRY_C [20];
        UCHAR ACTUAL_STOCK_LEVEL_C [12];
        UCHAR RECOMMENDE_STOCK_C [12];
        UCHAR CURRENCY_RESTRICT_INFO_C [80];
        UCHAR FOREIGN_INFO_C [80];
    } CURRENCIES [5];
    STRUCT
    {
        UCHAR TYPE_OF_CURRECY_D [3];
        UCHAR DENOMINATION_D [9];
        UCHAR STOCK_LEVEL_D [9];
    } CURRENCY_DENOMI [50];
} UPDATESTOCKORDER;
```

*Figure 23. Complex COMMAREA (Write)*

#### *Service Definition*

Figure 24 on page 91 shows the service definition for UpdateStockOrder.

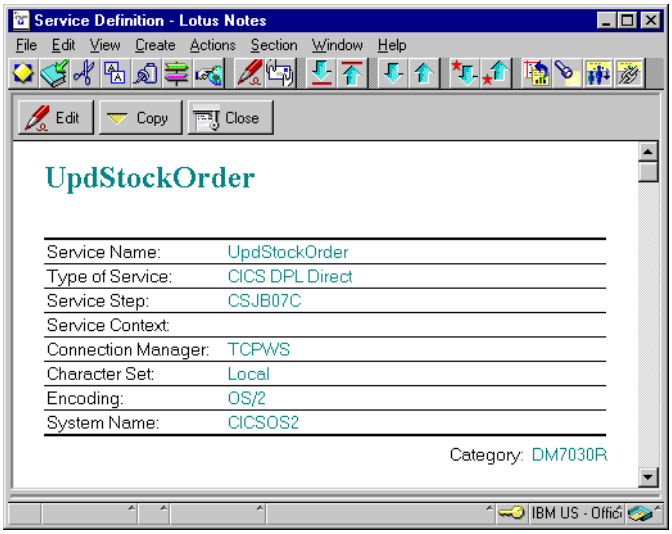



*Figure 24. Service Definition: Complex Message (Write)*

MQEI service UpdateStockOrder connects to CICS DPL program CSJB07C.

### *Message Definition*

Figure 25 shows MsgUpdateStockOrder message definition. This message is used by MQEI service UpdateStockOrder to exchange data with the enterprise CICS program. This message definition corresponds to the CICS COMMAREA.

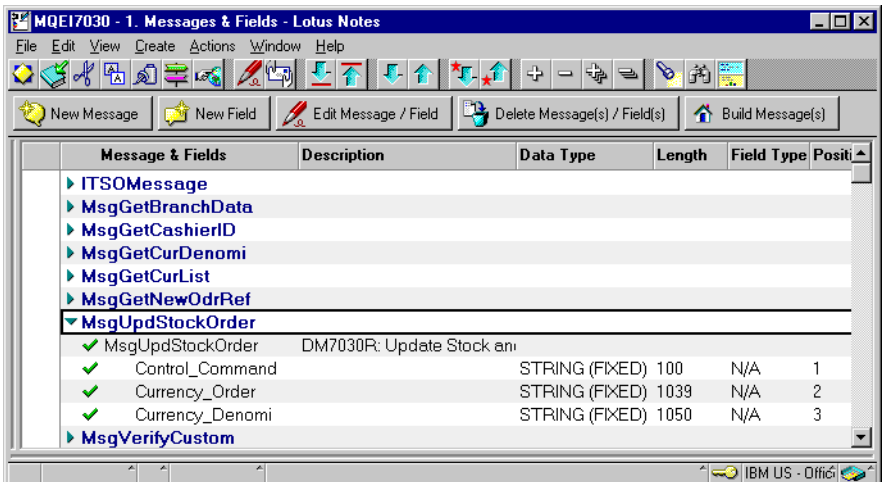



*Figure 25. Message Definition: Complex Message (Write)*

In MsgUpdateStockOrder, we created three message fields:

- Control_Command corresponds to the system data set automatically by the MQEI.
- Currency_Order corresponds to a cashier order of five currencies as input (see Figure 26). The Currency_Order length is 1039 bytes (207 x 5 + 4).
- Currency_Denomi corresponds to a list of 50 denominations for currencies. The Currency_Denomi length is 1050 bytes (21 x 50).

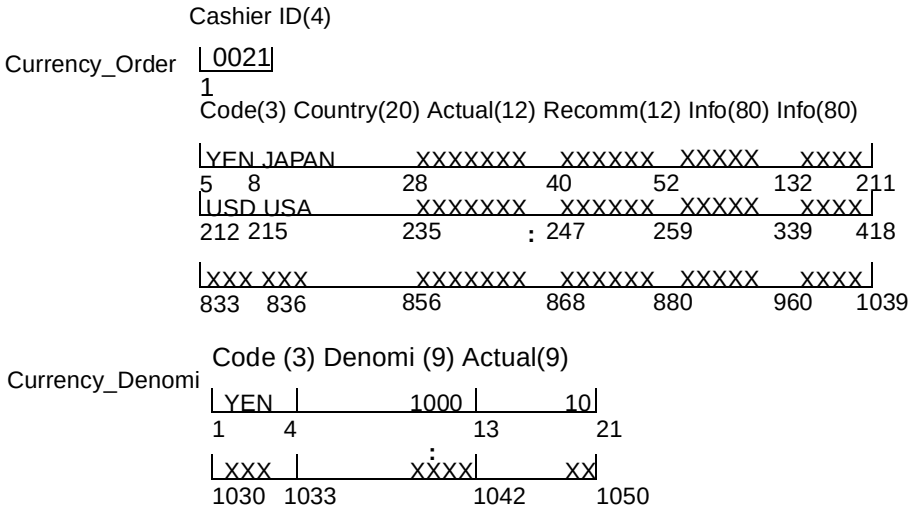



*Figure 26. Currency Order and Denomination Fields in MsgUpdStockOrder*

### *LotusScript Function*

In the LotusScript function, we performed the following steps:

1. We created EISession, EIService, EIMessage, EISendOptions, and EIReceiveOptions objects and set the currency order and denomination in the MQEI message:

```
Set MySession = New EISession
Set MyService = MySession.CreateService("UpdStockOrder")
Set MyMessage = MySession.CreateMessage("MsgUpdStockOrder")
Set MySendOptions = MySession.CreateSendOptions
Set MyReceiveOptions = MySession.CreateReceiveOptions
Mymessage.Currency_Order = Currency_Order
Mymessage.Currency_Denomi = Currency_Denomi
```

2. We connected to the enterprise service, using the MQEI service, sent the message, and waited for a reply:

```
Call MyService.Connect
```

```
Call MyService.SendMessage (MyMessage, MySendOptions)
MyReceiveOptions.Identifier = MySendOptions.Identifier
Do
    Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
    If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
        retcode = False
        Exit Do
    Else
        retcode  = True
        Exit Do
    End If
Loop
```

### 6.1.4 Benefits and Recommendations

The LotusScript application connects to CICS and carries out, according to the service requested in the Lotus Notes document, the unit of work through calls to the ECI. The benefits of using the CICS DPL direct service of the MQEI are:

- Coding is not required in LotusScript to handle CICS security; the MQEI handles security automatically.
- The ECI does not require screen scraping.
- Communication between LotusScript and CICS is simplified with the transfer of only one message (COMMAREA) between environments.

The MQEI requires that you map the MQEI message and the COMMAREA. With complex COMMAREAs that include substructures, we recommend merging them into one LotusScript string and parsing the string within the Lotus Notes application. In DOM98, we did not redesign the COMMAREAs. However, redesign may be a good option.

## 6.2 NotesPump

In CS92, DB2 access was always through a CICS transaction, even if the data was only read. However, as the branch office was also running a CICS server, this architecture satisfied the requirement to put the data as close to the user as possible.

In DOM98, the branch office server is a Domino server. Reference data still has to be located as close as possible to the user. Therefore, we used NotesPump to copy data from the enterprise DB2 to the branch office Domino server. The data copied is only read on Domino. Data updates are still performed directly on the DB2 master database through CICS transactions.

### 6.2.1 Administrator Database

System administrators manage NotesPump by using a Lotus Notes application called *NotesPump Administrator*. They create and populate Notes documents that are the instruction set carried out by the NotesPump server.

Figure 27 shows the DOM98 NotesPump environment.

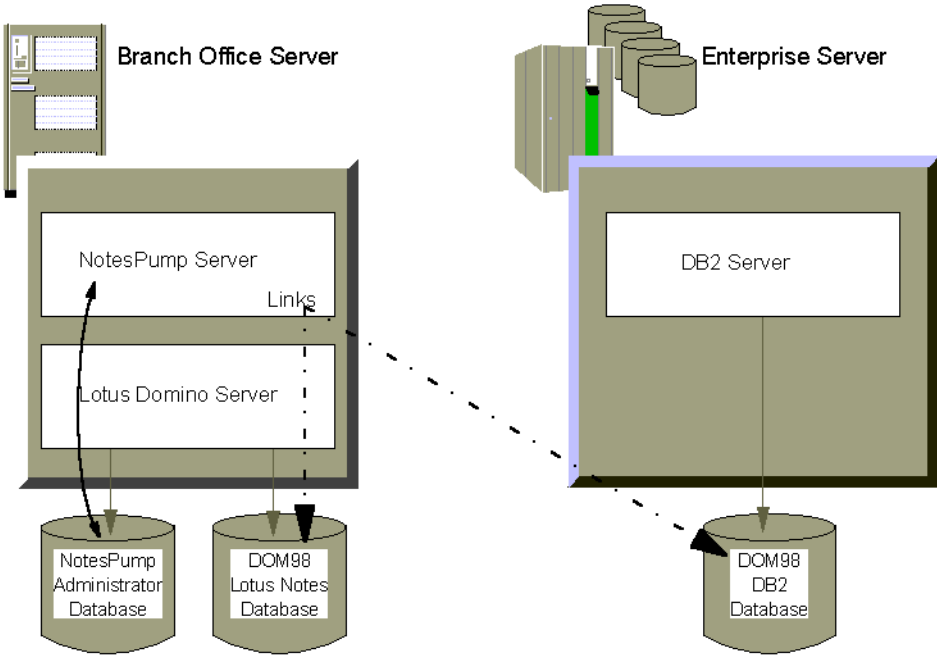



*Figure 27. DOM98 NotesPump Environment*

The branch office machine runs the NotesPump server. It also runs the Domino server that manages the DOM98 Lotus Notes database and the NotesPump administrator database. To connect NotesPump to the two external databases, we defined two link documents:

- Currency table, which defines the enterprise DB2 server managing the currency table
- Currency list, which defines the DOM98 Lotus Notes database and its server managing the currency list documents

### 6.2.2 Direct Transfer Activity

A Direct Transfer Activity document is created in the NotesPump Administrator database to copy the currency data from the DB2 database to the Domino environment.

Figure 28 shows the field mapping between DB2 and Domino defined in the Direct Transfer activity document.

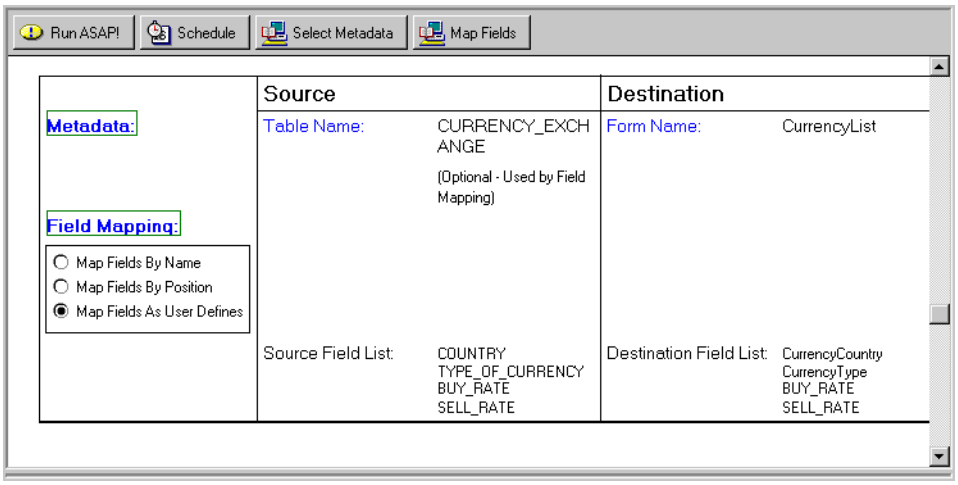



*Figure 28. Direct Transfer Activity Document*

Figure 29 on page 96 shows the currency list view in the DOM98 Lotus Notes database as the result of the direct transfer activity.

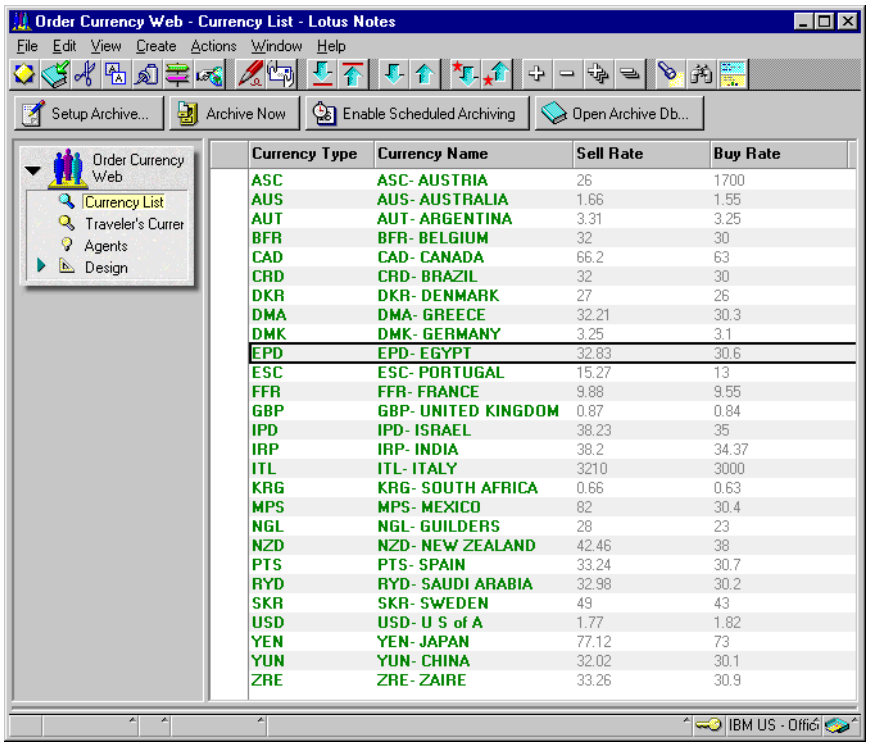



*Figure 29. Currency List View*

### 6.2.3 Realtime Notes Activity

As an alternative solution to a LotusScript program using the MQEI, we developed a lookup function, using the NotesPump Realtime Notes activity. Replacing existing CICS transactions, we used this activity to get currency and denomination lists directly from DB2.

#### 6.2.3.1 Description

A Realtime Notes activity provides synchronous access to NotesPump-supported data sources from a Notes application.

When a Realtime Notes activity has been created and enabled, it intercepts Notes database events, such as open, create, update, or delete of Notes documents and, based on the event, obtains “real-time” data from the data sources. Once a system administrator has created the Realtime Notes activity, the Notes application can open, create, update, or delete the data in the data source directly and transparently. Realtime Notes activity is

managed on the Domino server and supports Lotus Notes clients as well as Web browsers.

Network access to the data sources is handled by the Lotus Domino and NotesPump server machine. This machine contains the connectivity software for connecting to the data source. In DOM98, our Lotus Domino server connects to the DB2 server through the DB2 client application enabler (CAE). Lotus Notes clients using the Realtime Notes activity forms do not need any connectivity software to the data source.

For Realtime Notes activities to run properly, the Realtime Extension Manager library must be correctly set up and registered in the Domino server. In the NOTES.INI of your Domino server, add the following line:

```
EXTMGR_ADDINS = LNPEXT.DLL (for Windows NT & OS/2)
```

This extension manager is activated when Realtime Notes activity is started and is invoked when a specified form event (open, update, create, or delete) occurs in Lotus Notes or Web clients.

Figure 30 shows the architecture of the Realtime Notes activity.

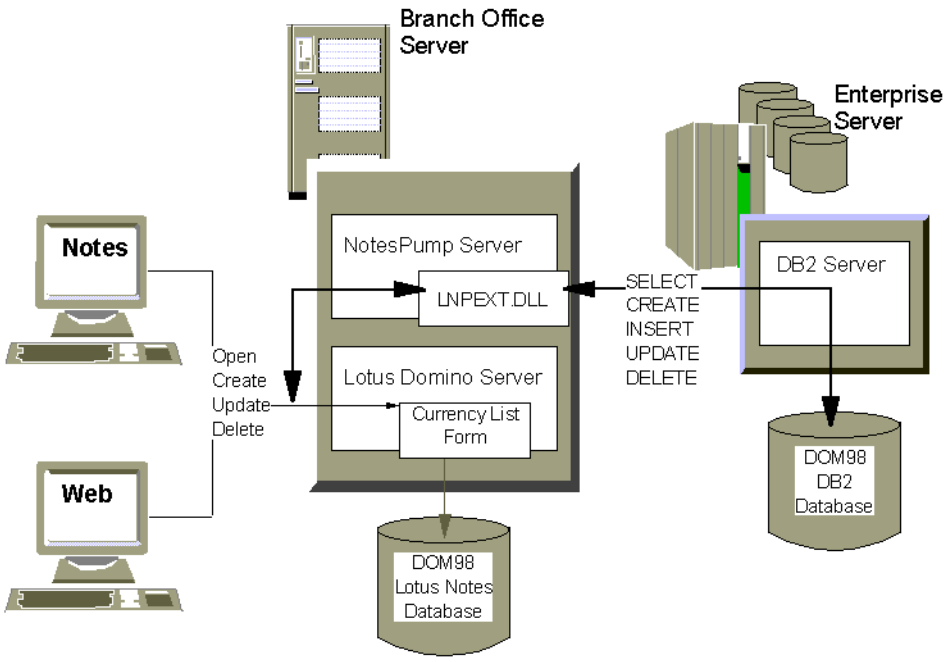



*Figure 30. Realtime Notes Activity Architecture*

### 6.2.3.2 Realtime Notes Activity Document

In DOM98, we created Currency documents using a Direct Transfer activity. However, to prevent copying data that rapidly becomes outdated, we did not transfer the actual and the recommended stock levels. Using the Realtime Notes activity, the application can access that data directly from DB2 without additional software and LotusScript programming. The application code is thus smaller as there is no need to create database access functions, and application development time is thus minimal.

Figure 31 shows the Realtime Notes activity definition.

**Links**

| Notes Database: | WebCurr.nsf |
|---|---|

| | External Link | |
|---|---|---|
| Link Name: | Currency Table | (DB2) (No Link Options) |

| | Default | Override |
|---|---|---|
| Server: | N/A | |
| Database: | BRANCHDB | |
| User Name: | CICSRS3 | |
| Password: | ****** | |

Password(s) NOT Encrypted

| | Notes | | External Link | |
|---|---|---|---|---|
| Metadata: | Form Name: | CurrencyList | Table Name: | CURRENCY_EXCHANGE |
| Field Mapping: | Notes Key List: | CurrencyType | External Link Key List: | TYPE_OF_CURRENCY |
| | Notes Field List: | BUY_RATE<br>SELL_RATE<br>ACT_STOCK_LEVEL<br>REC_STOCK_LEVEL<br>CURR_RES_INFO<br>FORGERY_INFO | External Link Field List: | BUY_RATE<br>SELL_RATE<br>ACT_STOCK_LEVEL<br>REC_STOCK_LEVEL<br>CURR_RES_INFO<br>FORGERY_INFO |

*Figure 31. Realtime Notes Activity Definition*

This activity is invoked when users open the CurrencyList form in the WebCurr.nsf database. Figure 32 on page 99 shows the settings for the Realtime Notes events.

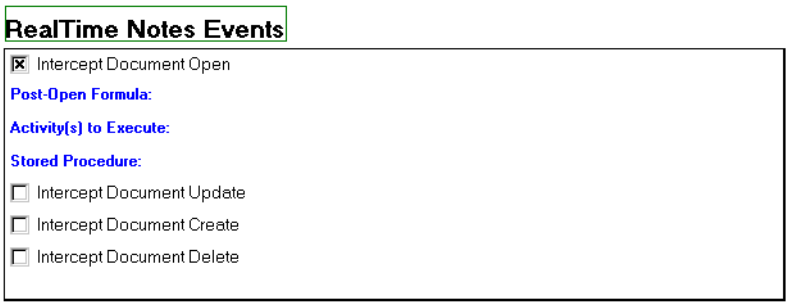



*Figure 32. Realtime Notes Events Settings*

When the Realtime Notes activity is not started, the form does not access the DB2 tables, and the actual and recommended stocks level fields are empty (see Figure 33).

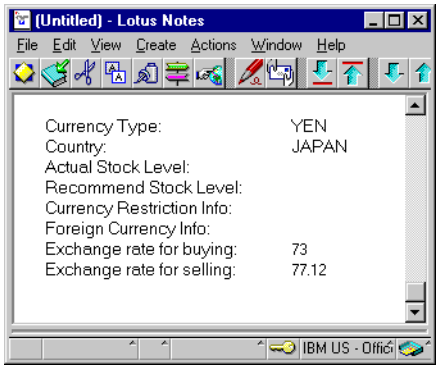



*Figure 33. Currency Form without Realtime Notes Activity*

As soon as the activity is started, the form displays the complete information (see Figure 34 on page 100).

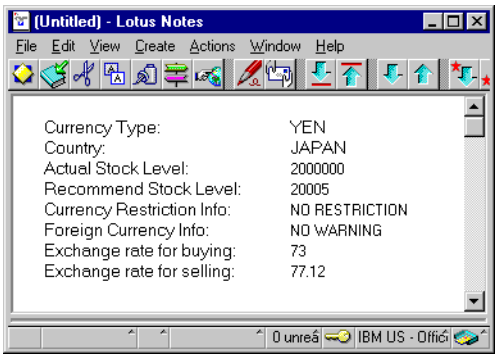



*Figure 34. Currency Form with Realtime Notes Activity*

The currency type, country, and exchanges rates for buying and selling are stored in the Notes document. Using the document currency type value as a key, the Realtime Notes activity fetches the accurate stock levels and information directly from the DB2 table when the document is opened.

### 6.2.4 Benefits and Recommendations

Using NotesPump you can easily set up the transfer of data from one environment to another. As you enter the copy parameters into a NotesPump document, you do not have to do any LotusScript programming. NotesPump provides a rich set of functions for activity scheduling and error handling.

Use NotesPump to copy data in your application if:

- Data volatility is low. Because you are working on a copy of the actual data, the changes must be infrequent. In our case, the name of the currency is very stable information.
- Copied data is accessed only for read. In that case, you can bypass the business transaction to access the data.
- The volume of transferred data is adequate for periodic copy.

Using NotesPump Realtime Notes activity, you can access from your Lotus Notes application DB2 data in real time. In DOM98, a Realtime Notes activity replaced the GetCurrencyList and GetDenominationList functions developed through the MQEI. It enabled us to avoid the 64 KB string limit of a LotusScript function. To use LotusScript, we would have had to modify the CICS program or change the code of the Notes application.

All Realtime Notes activity parameters are accessible from LotusScript programs using the NotesDocument class.

## 6.3 DOM98 Lotus Notes Database

In this section we explain the different components of the DOM98 Lotus Notes database.

Lotus Notes provides a complete IDE that enables you to create a variety of elements used in Notes applications, namely, forms, views, navigators, and agents.

### 6.3.1 CS92 Graphical User Interface

The CS92 GUI consisted of four main parts:

- Logon window, where the user signed on to enter the application
- Traveler's Currency window, which displayed the application main menu that presents all available options to the user
- Order Handling window, where the user could create, update, and delete an order
- Currency Specification window, where the user could select different currencies and quantities

### 6.3.2 Lotus Notes Graphical User Interface

In this section we describe the GUI that we created in DOM98 for Lotus Notes users.

***Logon Window***
We relied on Lotus Notes security mechanisms to identify users. Therefore, we did not create a specific logon function. However, to prevent access from unauthorized users, we carefully defined the ACL of our Lotus Notes database.

***Traveler's Currency Window: A View***
We used a view for the DOM98 main menu (see Figure 35 on page 102), from which users can select available operations through action buttons.

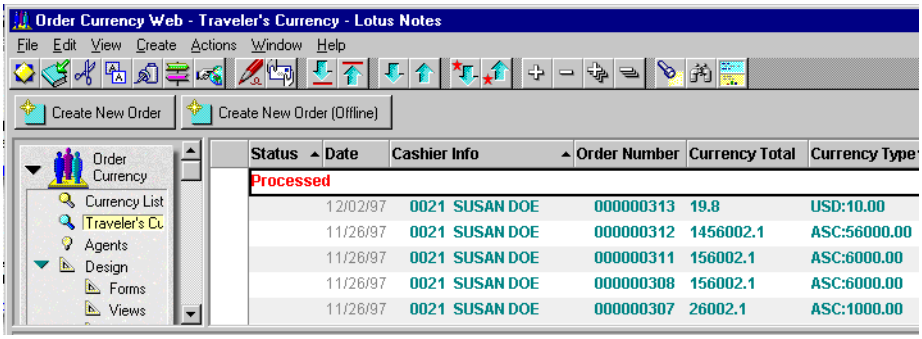



*Figure 35. Traveler's Currency View: Notes*

### *Order Handling Window: A Form*

When selecting the **Create New Order** button, the user is presented with the Order Handling form. Again, the user selects operations with action buttons, such as **Get Order Reference**, **Verify Account Number**, **Add New Currency**, and **Complete Order**. These buttons replace the drop-down menus used in CS92.

Figure 36 on page 103 shows the Currency Order Handling form.

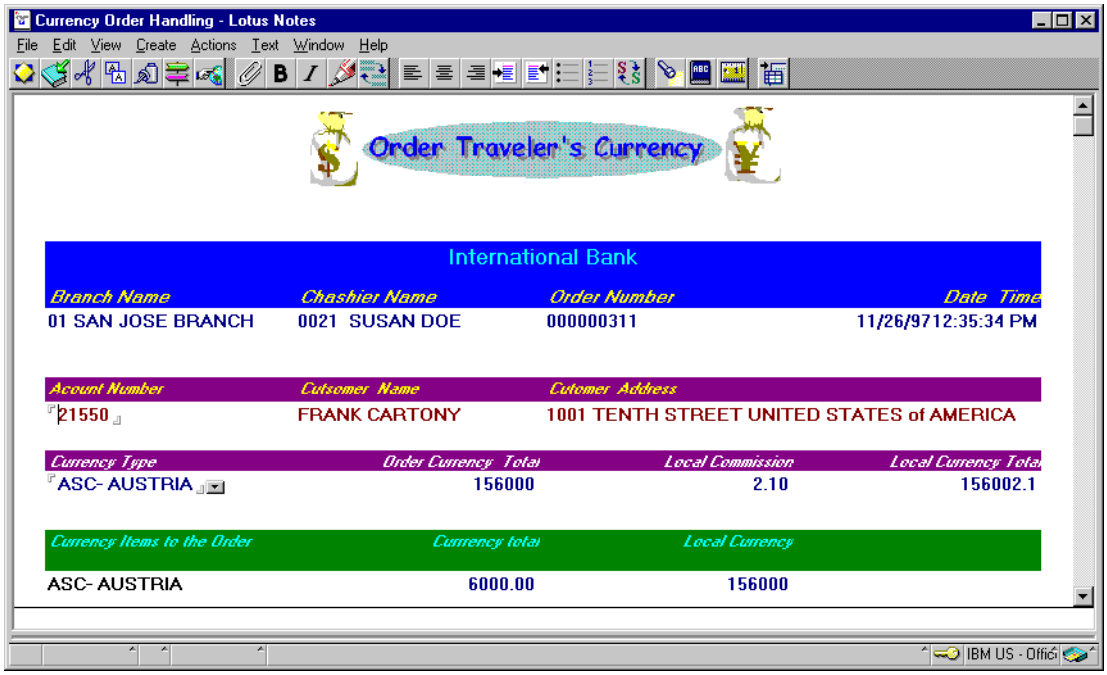



*Figure 36. Currency Order Handling Form: Notes*

Figure 37 shows a dialog box where users can select a currency.

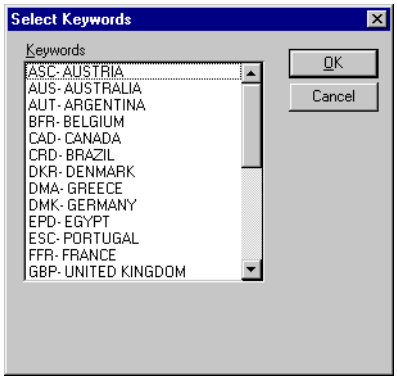



*Figure 37. Currency Selection*

### Currency Specification Window: A Form

In DOM98, when the user clicks on the **Add New Item** button, a response form is displayed. Although this form is a response document, it is opened as

a dialog to prevent users from changing its focus until they finish entering their data. We used the following DialogBox method:

```
Call uiWorkspace.Dialogbox("Currency Details", True, _
True, False, True,False, False, "Currency Specification Facility")
```

Figure 38 shows the specification form for the selected currency.

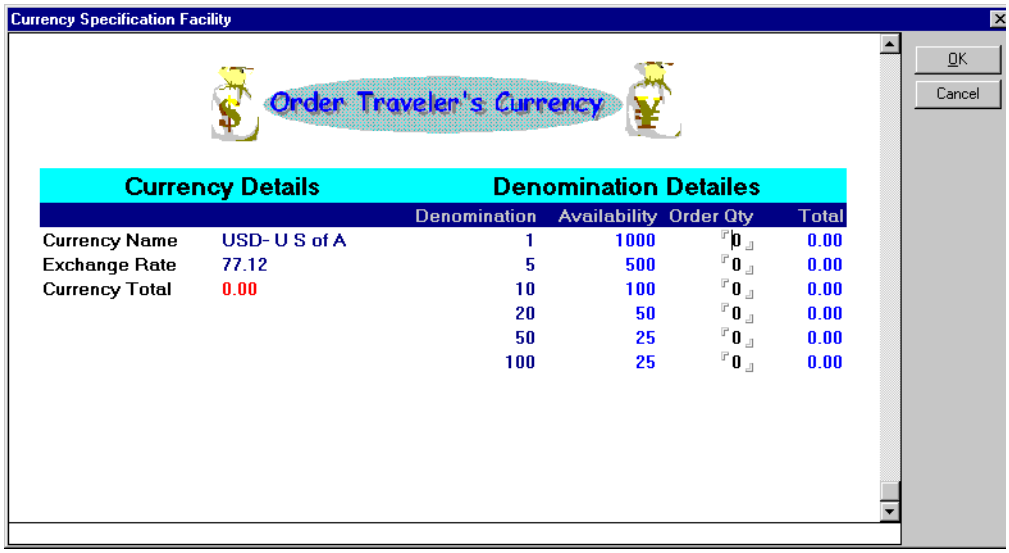



*Figure 38. Currency Specification Form: Notes*

After entering the quantities ordered, the user clicks on the **OK** button.

As we set *Automatically refresh fields* on the Default page of the form properties, all fields are automatically recalculated when the user modifies any input field or clicks on the **OK** button. Currency Total and each Denomination Total fields, defined as computed fields, are automatically calculated.

### 6.3.3 Web Browser Graphical User Interface

In this section we describe the GUI we created in DOM98 for Web users.

#### *Logon Window*

In the ACL of the DOM98 database, we set the default access to None. The Domino server public address book does not contain an entry for an anonymous user, and each Web user is defined with an HTTP password.

We used the Domino Web security to protect our application.

When connecting to DOM98, each Web user must first authenticate (see Figure 39).


*Figure 39. Web User Logon Window*

### *Traveler's Currency Window: A View*

DOM98 uses the same view as the Lotus Notes clients. Action buttons are displayed on the Web as well (see Figure 40).

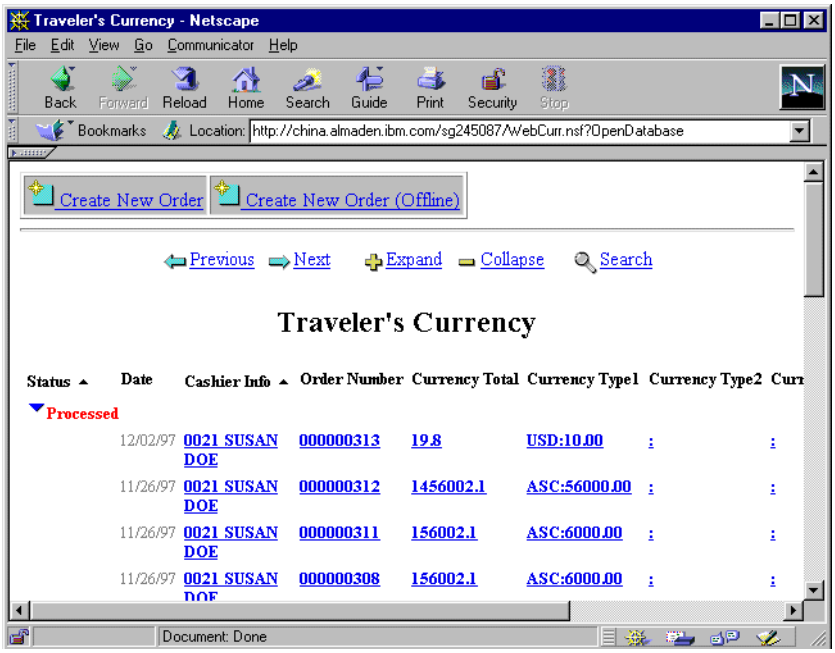



*Figure 40. Traveler's Currency View: Web*

### *Order Handling Window: A Form*

We used the same Order Handling form defined for Lotus Notes users. However, as the LotusScript code associated with the action button Click

event is not supported on the Web, we replaced action buttons with hotspot buttons. Domino Version 4.6 supports multiple hotspot buttons on the Web.

To display buttons according to the status of the application, we set the *Hide When* option of the button properties. Figure 41 shows how to hide the Get Customer Information button for users of Lotus Notes clients and when an order has already been created. This hide property is effective for the whole line. Therefore it may be necessary to create multiple button lines.

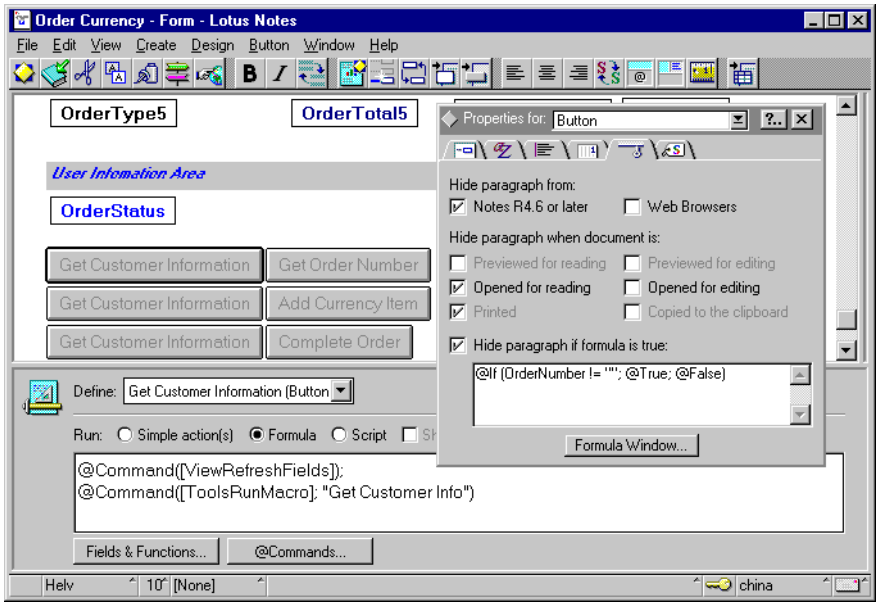



*Figure 41. Hide When Tab*

Figure 42 on page 107 shows the Currency Order Handling form displayed in a Web browser.


*Figure 42. Currency Order Handling Form: Web*

### Currency Specification Window: A Form

The Currency Specification form is used by a Lotus Notes client. It has features such as automatic field calculation and input field validation. On the Web, however, these functions are not available unless the page is submitted back to the server.

We created a Web-only Currency Details form based on the Notes-only Currency Specification form and used JavaScript to provide calculation and validation.

All Notes-only computed fields became input fields. To prevent users from entering values in those fields, we used a JavaScript procedure in the fields HTML attribute, as shown in Figure 43 on page 108.

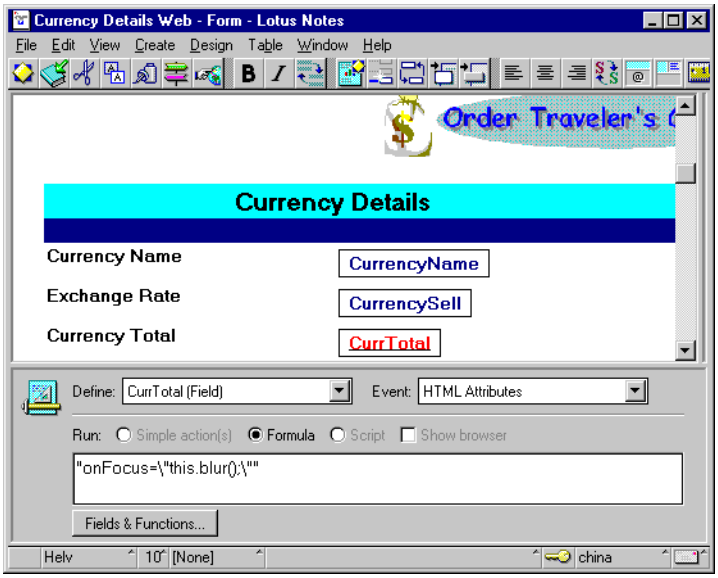



*Figure 43. JavaScript: HTML Attributes Event*

Figure 44 shows the JavaScript procedure used to calculate the total when a user clicks the **Submit** button or when the field loses its focus.

```
"onBlur=\" if(this.form.Avail_6.value-this.form.Order6.value>=0 &&
this.form.Order6.value>=0) {
    this.form.Total6.value=this.form.Denomi_6.value*this.form.Order6.value;
    this.form.Total_6.value=this.form.Denomi_6.value*this.form.Order6.value;
    this.form.CurrTotal.value=eval(this.form.Total_1.value)+
    eval(this.form.Total_2.value)+eval(this.form.Total_3.value)+
    eval(this.form.Total_4.value)+eval(this.form.Total_5.value)+
    eval(this.form.Total_6.value)+eval(this.form.Total_7.value)+
    eval(this.form.Total_8.value)+eval(this.form.Total_9.value)+
    eval(this.form.Total_10.value);
} else {
    alert(\\\'Invalid Input Order Quantity!\');
};
window.status=\'\'; \""
```

*Figure 44. JavaScript: Calculate Total Fields*

Figure 45 on page 109 shows the Currency Details form as displayed by a Web browser. The pop-up dialog window is generated by the JavaScript procedure when a user enters invalid values.

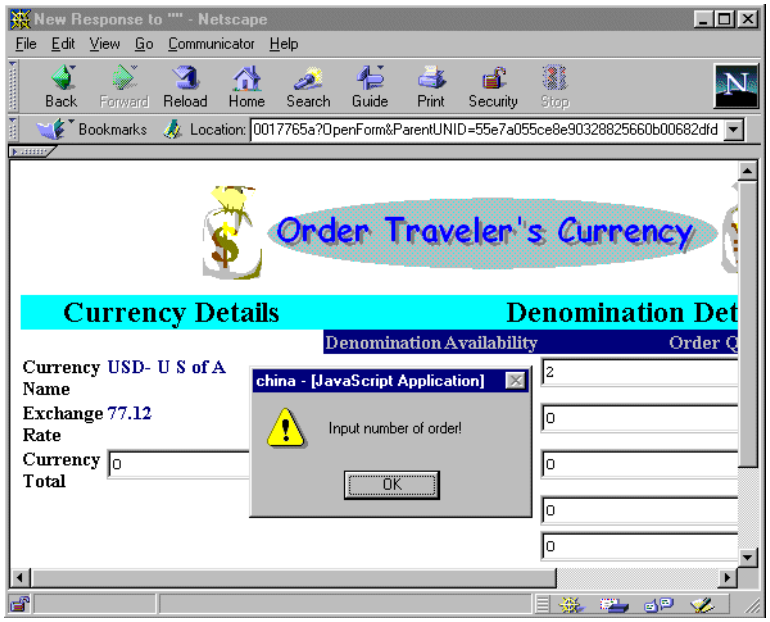



*Figure 45. Currency Details Form: Web.*

### 6.3.4 Script Library

All LotusScript functions using the MQEI are stored in a script library, called *OrderCurrency*, within the database. Any element of the application (form, view, field, or agent) can take advantage of these functions when a *Use* statement is entered in its Options section:

```
Use "OrderCurrency"
```

The results of these functions are returned in common variables, also defined in the Declarations section.

To be able to use identical LotusScript functions for Lotus Notes client and Web browser applications, we avoided using NotesUIDocument and NotesDocument classes in the application. Therefore, agents supporting Web and mobile Notes users can call these functions.

### 6.3.5 Supporting Lotus Notes Users

As DOM98 supporting Lotus Notes users runs on a Lotus Notes client, the application can take advantage of the NotesUIDocument classes. Therefore, we simply used two types of events of the Lotus Notes Object to access the enterprise services:

- In the PostOpen event, we gather all the data needed to perform the operation (branch and cashier data, and currency list). This event occurs when the document is opened for the user.
- In the Click event of any buttons available on the form, we process the user requested action (Get order reference, Verify account number, Add new item, Complete order).

Figure 46 shows the organization of DOM98 supporting Lotus Notes users.

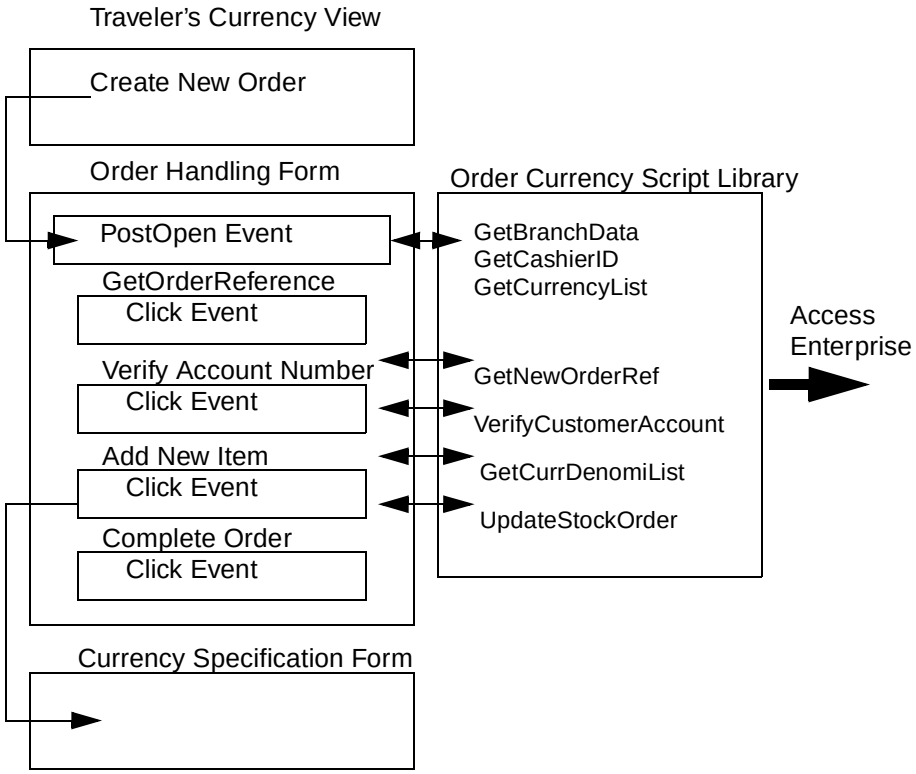



*Figure 46. Lotus Notes User Support*

### 6.3.6 Supporting Mobile Notes Users

Mobile users do not have a direct connection to the enterprise but replicate their information to the Domino server instead. Then, according to a schedule, a server agent processes all new orders to the enterprise service.

Use of a scheduled server agent can also be a valid backup solution, processing the orders stored locally whenever the Domino server or the enterprise server are down.

Figure 47 shows the organization of DOM98 supporting mobile users.

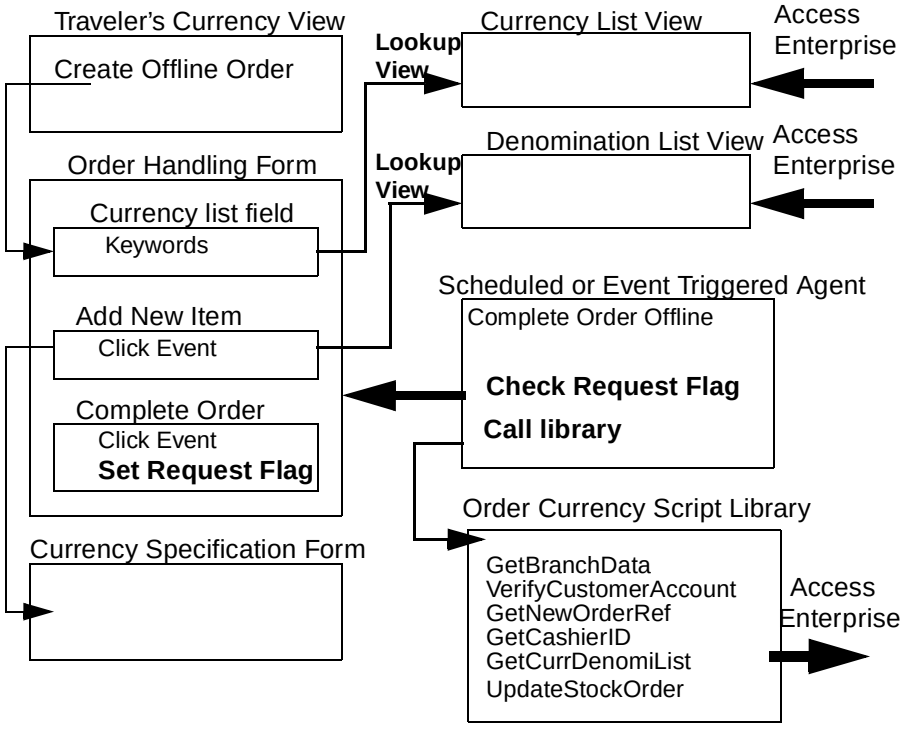



*Figure 47. Mobile Notes User Support*

As there is no connection to the enterprise server, all data must be local to the user's workstation. Using the NotesPump direct transfer activity, we copied the currency list from the enterprise server to the Domino database and then replicated on the user's workstation. In that offline mode, the Currency List view is a simple view displaying currency type and name. Figure 48 on page 112 shows how to use the Currency List view with a keywords field.

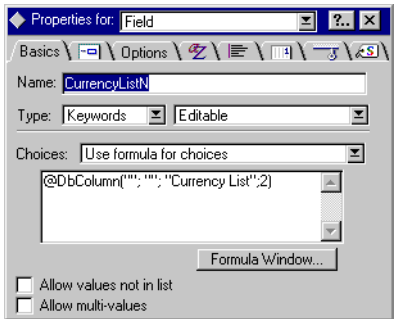



*Figure 48. Using the Currency List View*

Of course, the duplicated data must be periodically refreshed from the enterprise server, using the NotesPump direct transfer activity or a DB2LSX program.

The scheduled server agent processes documents that have the flag set to "Request Process." It invokes the script library functions, processes the document with the enterprise service, resets the flag, and stores the document on the Domino server. Before invoking the enterprise server, we used the ComputeWithForm method of the NotesDocument class to recalculate all values of the form.

### 6.3.7 Supporting Web Users

In this configuration, we used an event and formulas to access the enterprise service:

- From the Web, the only available event is the *WebQueryOpen* event, and it occurs before the form is opened. We used this event to gather all data needed to perform the operation (branch and cashier data, and currency list).
- Domino evaluates formulas when the Web user clicks on the associated buttons. On the Domino server, each formula uses an @Command function to start the corresponding Web agent (Get order reference, Verify account number, Add new item, Complete order). The Web client waits until the agent finishes and prints back the results to the Web.

Figure 49 on page 113 shows the organization of DOM98 supporting Web users.

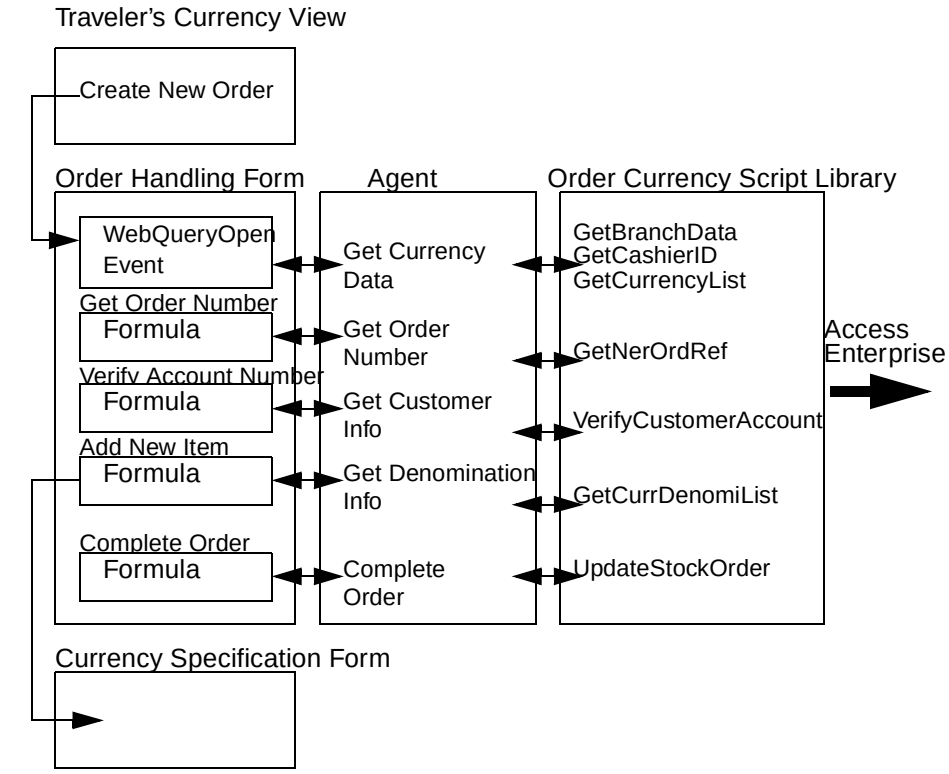



*Figure 49. Web User Support*

# Appendix A. Script Library

In this appendix we list the code used for the script library.

## A.1 Option and Declaration

```
(Option)
Option Public
' Include Notes constants file (defines MB_OK etc)
%INCLUDE "lsconst.lss"
' Load the MQSeries Enterprise Integrator library
Uselsx "*eilsx"
(Declarations)
Dim CASHIER_ID As String
Dim CASHIER_NAME As String
Dim BANK_NAME As String
Dim BRANCH_DATA As String
Dim BRANCH_RATE As String
Dim CUSTOMER_NUM As String
Dim CUSTOMER_NAME As String
Dim CUSTOMER_ADDR As String
Dim ORDER_NUMBER As String
Dim ORDER_ADDRESS As String
Dim CUR_TYPE As String
Dim CUR_CODE As String
Dim CUR_ACTS As String
Dim CUR_RECM As String
Dim CUR_INFO As String
Dim CUR_FORG As String
Dim EXCHANGE_SELL As String
Dim EXCHANGE_BUY As String
Dim DENOMI(10) As String
Dim DORDER(10) As String
Dim DSTOCK(10) As String
Dim COMP_CURR As String
Dim COMP_DENOMI As String
Dim CUST_ORDER As String
Dim CUST_DENOMI As String
Dim CUST_CURR As String
Dim SERVICE_NAME As String
Dim MESSAGE_NAME As String
Const TABLE_SIZE = 207
Const MAX_TABLE = 30
Const MAX_DENOMI = 10
Const DENOMI_SIZE = 21
'   Define Enterprise Integrator objects.
Dim MySession As EISession
Dim MyService As EIService
Dim MyMessage As EIMessage
Dim MySendOptions As EISendOptions
Dim MyReceiveOptions As EIReceiveOptions
```

## A.2 GetCashierID Function

```
'*************************************************
' Function Name Get Cashier ID
' Input Data          Cashier Name
' Output Data       Cashier ID
' Service Name    GetCashierID
' Message Name MsgGetCashierID
'*************************************************
Function GetCashierID (CashierName As String) As Long
     Const LEN_NAME  = 40
     Const FIELD_ID = LEN_NAME + 1
     Const LEN_ID = 40
     Dim rcvdata As String
     Dim retcode As Long
     SERVICE_NAME = "GetCashierID"
     MESSAGE_NAME = "MsgGetCashierID"
' Set up MQEI Services
     Set MySession = New EISession
     Set MyService = MySession.CreateService(SERVICE_NAME)
     Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
     Set MySendOptions = MySession.CreateSendOptions
     Set MyReceiveOptions = MySession.CreateReceiveOptions
     MySendOptions.UnitofWork = EIUOW_ONLY
     MyReceiveOptions.WaitType = EIWT_WAIT
     MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
     Call MyService.Connect
' Send Request
     MyMessage.Cashier_Name = CashierName
     Call MyService.SendMessage (MyMessage, MySendOptions)
' Receive  Reply
     MyReceiveOptions.Identifier = MySendOptions.Identifier
     Do
          Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
          If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
               retcode = False
               Exit Do
          Else
               retcode  = True
               Exit Do
          End If
     Loop
'Return to Caller
     rcvdata = MyMessage.Cashier_Name
     CASHIER_ID= Trim(Mid$(rcvdata,FIELD_ID,LEN_ID))
     CASHIER_NAME=Trim(Left$(rcvdata, LEN_NAME))
     GetCashierID = retcode
End Function
```

## A.3 GetNewOdrRef Function

```
'************************************************
' Function Name Get New Order Reference Number
' Input Data          None
' Output Data       Order Number
' Service Name    GetNewOdrRef
' Message Name MsgGetNewOdrRef
'************************************************
Function GetNewOdrRef As Long
     Const LEN_CUSTOMER  = 9
     Const FIELD_BRANCH = LEN_CUSTOMER + 1
     Const LEN_BRANCH = 9
     Const FIELD_CASHIER = FIELD_BRANCH + LEN_BRANCH + 1
     Const LEN_CASHIER = 9
     Dim rcvdata As String
     Dim retcode As Long
     SERVICE_NAME = "GetNewOdrRef"
     MESSAGE_NAME = "MsgGetNewOdrRef"
' Set up MQEI Services
     Set MySession = New EISession
     Set MyService = MySession.CreateService(SERVICE_NAME)
     Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
     Set MySendOptions = MySession.CreateSendOptions
     Set MyReceiveOptions = MySession.CreateReceiveOptions
     MySendOptions.UnitofWork = EIUOW_ONLY
     MyReceiveOptions.WaitType = EIWT_WAIT
     MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
     Call MyService.Connect
' Send Request
     ' No Input Message of this function
     Call MyService.SendMessage (MyMessage, MySendOptions)
' Recive  Reply
     MyReceiveOptions.Identifier = MySendOptions.Identifier
     Do
          Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
          If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
               retcode = False
               Exit Do
          Else
               retcode = True
               Exit Do
          End If
     Loop
'Return to Caller
     rcvdata = MyMessage.Customer_Order
     ORDER_NUMBER =Trim(rcvdata)
     GetNewOdrRef = retcode
End Function
```

## A.4 GetCurrencyList Function

```
'*************************************************
' Function Name Get CurrencyList
' Input Data          Cashier ID
' Output Data       CurrencyList
' Service Name    GetCurList
' Message Name MsgGetCurList
'*************************************************
Function GetCurrencyList (CashierID As String) As Long
     Dim curtable(MAX_TABLE) As String
     Dim rcvdata As String
     Dim retcode As Long
     Dim count As Integer
     Const LEN_TYPE = 3
     Const LEN_CODE = 20
     Const LEN_ACTS = 12
     Const LEN_RECM = 12
     Const LEN_INFO = 80
     Const LEN_FORG = 80
     Const FIELD_CODE = LEN_TYPE + 1
     Const FIELD_ACTS = FIELD_CODE + LEN_CODE
     Const FIELD_RECM = FIELD_ACTS + LEN_ACTS
     Const FIELD_INFO = FIELD_RECM + LEN_RECM
     Const FIELD_FORG = FIELD_INFO + LEN_INFO
     SERVICE_NAME = "GetCurList"
     MESSAGE_NAME = "MsgGetCurList"
' Set up MQEI Services
     Set MySession = New EISession
     Set MyService = MySession.CreateService(SERVICE_NAME)
     Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
     Set MySendOptions = MySession.CreateSendOptions
     Set MyReceiveOptions = MySession.CreateReceiveOptions
     MySendOptions.UnitofWork = EIUOW_ONLY
     MyReceiveOptions.WaitType = EIWT_WAIT
     MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
     Call MyService.Connect
' Send Request
     'MyMessage.Cashier_ID = Space(LEN_ID)+CashierName + (0)
     'MyMessage.Cashier_Name = CashierName
     Call MyService.SendMessage (MyMessage, MySendOptions)
' Recive  Reply
     MyReceiveOptions.Identifier = MySendOptions.Identifier
     Do
          Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
          If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
               retcode = False
               Exit Do
          Else
               retcode  = True
               Exit Do
          End If
     Loop
' Return to Caller
     rcvdata = MyMessage.Currency_List
     For count = 0 To MAX_TABLE-1
          curtable(count) = Mid$(rcvdata, (count)*TABLE_SIZE + 1, TABLE_SIZE)
     Next
     For count = 0 To MAX_TABLE-1
          If Trim(Left$(curtable(count), LEN_TYPE)) <>"" Then
             CUR_TYPE = CUR_TYPE + Left$(curtable(count), LEN_TYPE) + " - "
```

```
            CUR_TYPE = CUR_TYPE + Trim(Mid$(curtable(count), FIELD_CODE, LEN_CODE)) +";"
            CUR_RECM = CUR_RECM + Trim(Mid$(curtable(count), FIELD_RECM, LEN_RECM)) +";"
            CUR_ACTS = CUR_ACTS + Trim(Mid$(curtable(count), FIELD_ACTS, LEN_ACTS)) +";"
            CUR_INFO = CUR_INFO + Trim(Mid$(curtable(count), FIELD_INFO, LEN_INFO)) +";"
            CUR_FORG = CUR_FORG + Trim(Mid$(curtable(count), FIELD_FORG, LEN_FORG)) +";"
        End If
    Next
    GetCurrencyList = retcode
End Function
```

## A.5 VerifyCustomerAccount Function

```
' ************************************************
' Function Name Verify CustomerAccount
' Input Data          Account Number
' Output Data       Account  Name, Account Address
' Service Name    VerifyCustom
' Message Name MsgVerifyCustom
'************************************************
Function VerifyCustomerAccount (AccountNum As String) As Long
    Const LEN_NUM  = 5
    Const FIELD_NAME = LEN_NUM + 1
    Const LEN_NAME = 40
    Const FIELD_ADDR = LEN_NUM + LEN_NAME + 1
    Const LEN_ADDR = 80
    Dim rcvdata As String
    Dim retcode As Long
    SERVICE_NAME = "VerifyCustom"
    MESSAGE_NAME = "MsgVerifyCustom"
' Set up MQEI Services
    Set MySession = New EISession
    Set MyService = MySession.CreateService(SERVICE_NAME)
    Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
    Set MySendOptions = MySession.CreateSendOptions
    Set MyReceiveOptions = MySession.CreateReceiveOptions
    MySendOptions.UnitofWork = EIUOW_ONLY
    MyReceiveOptions.WaitType = EIWT_WAIT
    MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
    Call MyService.Connect
' Send Request
    MyMessage.Customer_Info = AccountNum
    Call MyService.SendMessage (MyMessage, MySendOptions)
' Recive  Reply
    MyReceiveOptions.Identifier = MySendOptions.Identifier
    Do
        Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
        If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
            retcode = False
            Exit Do
        Else
            retcode  = True
            Exit Do
        End If
    Loop
' Return to Caller
    rcvdata = MyMessage.Customer_Info
    CUSTOMER_NAME= Mid$(rcvdata,FIELD_NAME, LEN_NAME)
    CUSTOMER_ADDR= Mid$(rcvdata, FIELD_ADDR, LEN_ADDR)
    VerifyCustomerAccount = retcode
End Function
```

## A.6 GetBranchData Function

```
'************************************************
' Function Name Get Branch Data and Commission Rate
' Input Data          None
' Output Data       Branch Data Details
' Service Name    GetBranchData
' Message Name MsgGetBranchData
'************************************************
Function GetBranchData As Long
     Const LEN_NUM  = 2
     Const LEN_NAME = 40
     Const LEN_ADDR = 80
     Const LEN_RATE = 4
     Const FIELD_NAME = LEN_NUM + 1
     Const FIELD_ADDR = FIELD_NAME + LEN_NAME
     Const FIELD_RATE = FIELD_ADDR + LEN_ADDR
     Dim rcvdata As String
     Dim retcode As Long
     SERVICE_NAME = "GetBranchData"
     MESSAGE_NAME = "MsgGetBranchData"
' Set up MQEI Services
     Set MySession = New EISession
     Set MyService = MySession.CreateService(SERVICE_NAME)
     Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
     Set MySendOptions = MySession.CreateSendOptions
     Set MyReceiveOptions = MySession.CreateReceiveOptions
     MySendOptions.UnitofWork = EIUOW_ONLY
     MyReceiveOptions.WaitType = EIWT_WAIT
     MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
     Call MyService.Connect
' Send Request
     Call MyService.SendMessage (MyMessage, MySendOptions)
' Recive  Reply
     MyReceiveOptions.Identifier = MySendOptions.Identifier
     Do
          Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
          If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
               retcode = False
               Exit Do
          Else
               retcode = True
               Exit Do
          End If
     Loop
' Return to Caller
     rcvdata = MyMessage.Branch_Data
     BRANCH_DATA =Left$(rcvdata, LEN_NUM) + " " + Mid$(rcvdata, FIELD_NAME, LEN_NAME)
     BRANCH_RATE =Trim(Mid$(rcvdata,FIELD_RATE,LEN_RATE))
     GetBranchData = retcode
End Function
```

## A.7 GetCurrDenomiList Function

```
'************************************************
' Function Name Get Currency Denomination List
' Input Data          CurrencyType
' Output Data       Currency Denomination List
' Service Name    GetCurDenomi
' Message Name MsgGetCurDenomi
'************************************************
Function GetCurrDenomiList (CurrencyType As String) As Long
     Dim denomitable(MAX_DENOMI) As String
     Dim rcvdata As String
     Dim retcode As Long
     Dim count As Integer
     Const LEN_TYPE = 3
     Const LEN_BUY = 7
     Const LEN_SELL = 7
     Const FIELD_BUY = LEN_TYPE + 1
     Const FIELD_SELL = FIELD_BUY + LEN_BUY
     Const FIELD_ARRAY = FIELD_SELL + LEN_SELL
     Const LEN_TYPE1 = 3
     Const LEN_LIST = 9
     Const LEN_STOCK = 12
     Const FIELD_LIST = LEN_TYPE1 + 1
     Const FIELD_STOCK = FIELD_LIST + LEN_LIST
     SERVICE_NAME = "GetCurDenomi"
     MESSAGE_NAME = "MsgGetCurDenomi"
' Set up MQEI Services
     Set MySession = New EISession
     Set MyService = MySession.CreateService(SERVICE_NAME)
     Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
     Set MySendOptions = MySession.CreateSendOptions
     Set MyReceiveOptions = MySession.CreateReceiveOptions
     MySendOptions.UnitofWork = EIUOW_ONLY
     MyReceiveOptions.WaitType = EIWT_WAIT
     MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
     Call MyService.Connect
' Send Request
     MyMessage.Denomi_List = CurrencyType
     Call MyService.SendMessage (MyMessage, MySendOptions)
' Recive  Reply
     MyReceiveOptions.Identifier = MySendOptions.Identifier
     Do
          Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
          If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
               retcode = False
                    Exit Do
               Else
               retcode  = True
               Exit Do
          End If
     Loop
' Return to Caller
     rcvdata = MyMessage.Denomi_List
     EXCHANGE_BUY = Mid$(rcvdata,FIELD_BUY,LEN_BUY)
     EXCHANGE_SELL= Mid$(rcvdata,FIELD_SELL,LEN_SELL)
     rcvdata = Mid$(rcvdata, FIELD_ARRAY, MAX_DENOMI*DENOMI_SIZE)
     For count = 1 To MAX_DENOMI
          denomitable(count) = Mid$(rcvdata,  (count-1)*DENOMI_SIZE + 1, DENOMI_SIZE)
     Next
```

```
     For count = 1 To MAX_DENOMI
          DENOMI(count) = Trim(Mid$(denomitable(count), FIELD_LIST, LEN_LIST))
          DSTOCK(count) = Trim(Mid$(denomitable(count), FIELD_STOCK,LEN_STOCK))
     Next
     GetCurrDenomiList = retcode
End Function
```

## A.8 UpdateStockOrder Function

```
'************************************************
' Function Name UpdateStockOrder
' Input Data          Currency Order,  Currency Denomination
' Output Data       None
' Service Name    UpdStockOrder
' Message Name MsgUpdStockOrder
'************************************************
Function UpdateStockOrder (Currency_Order As String,  Currency_Denomi As String)
     Dim retcode As Long
     SERVICE_NAME = "UpdStockOrder"
     MESSAGE_NAME = "MsgUpdStockOrder"
' Set up MQEI Services
     Set MySession = New EISession
     Set MyService = MySession.CreateService(SERVICE_NAME)
     Set MyMessage = MySession.CreateMessage(MESSAGE_NAME)
     Set MySendOptions = MySession.CreateSendOptions
     Set MyReceiveOptions = MySession.CreateReceiveOptions
     MySendOptions.UnitofWork = EIUOW_ONLY
     MyReceiveOptions.WaitType = EIWT_WAIT
     MyReceiveOptions.WaitInterval = 1000
' Connect MQEI Services
     Call MyService.Connect
' Send Request
     Mymessage.Currency_Order = Currency_Order
     Mymessage.Currency_Denomi = Currency_Denomi
     Call MyService.SendMessage (MyMessage, MySendOptions)
' Recive  Reply
     MyReceiveOptions.Identifier = MySendOptions.Identifier
     Do
          Call MyService.ReceiveMessage (MyMessage, MyReceiveOptions)
          If MyService.ReasonCode = EIRC_NO_MESSAGE_AVAILABLE Then
               retcode = False
               Exit Do
          Else
               retcode = True
               Exit Do
          End If
     Loop
     UpdateStockOrder = retcode
End Function
```

# Appendix B. Agents

In this appendix we list the code used for the agents.

## B.1 Add All Currency Data Agent

```
(Option)
Option Public
Use "OrderCurrency"

Sub Initialize
     Dim  session As New  NotesSession
     Dim db As NotesDatabase
     Dim doc As NotesDocument
     Dim account As String
     Dim customer As String
     Set db = session.CurrentDatabase
     Set doc = session.DocumentContext
     If doc.isnewnote = False Then  Exit Sub
'*************************************************************************
' *Get Branch Data Section
'*************************************************************************
     Call GetBranchData()
     Call doc.ReplaceItemValue ("BranchName", BRANCH_DATA)
     Call doc.ReplaceItemValue ("BranchCommRate", Cstr(BRANCH_RATE))
'*************************************************************************
' *Get Cashier ID Section
'*************************************************************************
     If  GetCashierID (doc.GetItemValue ("CashierName")(0)) = False Then
          Exit Sub
     End If
     If CASHIER_ID = "" Then
          Exit Sub
     Else
          Call doc.ReplaceItemValue ("CashierID", CASHIER_ID)
          Call doc.ReplaceItemValue ("CashierName",CASHIER_NAME)
          Call doc.ReplaceItemValue ("CashierData", CASHIER_ID + "  "  + _
                                     CASHIER_NAME)
     End If
'*************************************************************************
' *Get Currency List Section
'*************************************************************************
     Call GetCurrencyList(CASHIER_ID)
     Call doc.ReplaceItemValue ("CurrencyList", CUR_TYPE)
     Call doc.ReplaceItemValue ("ActualStock",CUR_ACTS)
     Call doc.ReplaceItemValue ("RecomStock", CUR_RECM)
     Call doc.ReplaceItemValue ("CurrencyInfo",CUR_INFO)
     Call doc.ReplaceItemValue ("ForgeryInfo",CUR_FORG)
End Sub
```

## B.2 Add New Currency Agent

```
Sub Initialize
     Dim session As New NotesSession
     Dim doc As NotesDocument
     Dim parent As NotesDocument
     Dim denomiorder As String
```

```
Dim currorder As String
Dim num As Integer
Dim localtotal As String
Dim order(10) As String
Dim denomi(10) As String
Dim demomilist As String

Set db = session.CurrentDatabase
Set doc = Session.DocumentContext
Set parent = db.GetDocumentByUNID(doc.ParentDocumentUNID )

denomi(1) = doc.GetItemValue ("Denomi1")(0)
denomi(2) = doc.GetItemValue ("Denomi2")(0)
denomi(3) = doc.GetItemValue ("Denomi3")(0)
denomi(4) = doc.GetItemValue ("Denomi4")(0)
denomi(5) = doc.GetItemValue ("Denomi5")(0)
denomi(6) = doc.GetItemValue ("Denomi6")(0)
denomi(7) = doc.GetItemValue ("Denomi7")(0)
denomi(8) = doc.GetItemValue ("Denomi8")(0)
denomi(9) = doc.GetItemValue ("Denomi9")(0)
denomi(10) = doc.GetItemValue ("Denomi10")(0)
order(1) =  doc.GetItemValue("Order1")(0)
order(2) =  doc.GetItemValue("Order2")(0)
order(3) =  doc.GetItemValue("Order3")(0)
order(4) =  doc.GetItemValue("Order4")(0)
order(5) =  doc.GetItemValue("Order5")(0)
order(6) =  doc.GetItemValue("Order6")(0)
order(7) =  doc.GetItemValue("Order7")(0)
order(8) =  doc.GetItemValue("Order8")(0)
order(9) =  doc.GetItemValue("Order9")(0)
order(10) =  doc.GetItemValue("Order10")(0)

For count =1 To 10
    denomiorder = denomiorder +
              Left(parent.GetItemValue("CurrencyListN")(0),3)  +_
              Right(Space(9) + denomi(count),9) + Right(Space(9) + _
              order(count),9)
Next
currorder = parent.GetItemValue("CashierID")(0) +_
            Left(parent.GetItemValue ("CurrencyListN")(0),3) +_
            Space(20) + Right(Space(12)+ _
            doc.GetItemValue("CurrTotal")(0),12) + Space(176)

num = Cint(parent.GetItemValue( "CurrentOrder" )(0))
Select  Case num
Case  1:
     Call parent.ReplaceItemValue ("OrderType1", _
          parent.GetItemValue("CurrencyListN"))
     Call parent.ReplaceItemValue ("OrderTotal1",_
          doc.GetItemValue("CurrTotal")(0) + ".00")
     Call parent.ReplaceItemValue ("OrderCurrency1", currorder)
     Call parent.ReplaceItemValue ("OrderDenomiQty1", denomiorder)
Case 2:
     Call parent.ReplaceItemValue ("OrderType2",_
          parent.GetItemValue("CurrencyListN"))
     Call parent.ReplaceItemValue ("OrderTotal2",_
          doc.GetItemValue("CurrTotal")(0) + ".00")
     Call parent.ReplaceItemValue ("OrderCurrency2", currorder)
     Call parent.ReplaceItemValue ("OrderDenomiQty2", denomiorder)
Case 3:
     Call parent.ReplaceItemValue ("OrderType3",_
          parent.GetItemValue("CurrencyListN"))
     Call parent.ReplaceItemValue ("OrderTotal3",_
```

```
            doc.GetItemValue("CurrTotal")(0) + ".00")
        Call parent.ReplaceItemValue ("OrderCurrency3", currorder)
        Call parent.ReplaceItemValue ("OrderDenomiQty3", denomiorder)
    Case 4:
        Call parent.ReplaceItemValue ("OrderType4",_
            parent.GetItemValue("CurrencyListN"))
        Call parent.ReplaceItemValue ("OrderTotal4",_
            doc.GetItemValue("CurrTotal")(0) + ".00")
        Call parent.ReplaceItemValue ("OrderCurrency4", currorder)
        Call parent.ReplaceItemValue ("OrderDenomiQty4", denomiorder)
    Case 5:
        Call parent.ReplaceItemValue ("OrderType5",_
            parent.GetItemValue("CurrencyListN"))
        Call parent.ReplaceItemValue ("OrderTotal5",_
            doc.GetItemValue("CurrTotal")(0) + ".00")
        Call parent.ReplaceItemValue ("OrderCurrency5", currorder)
        Call parent.ReplaceItemValue ("OrderDenomiQty5", denomiorder)
    Case Else
        Print  "No more Entry"
        Exit Sub
    End Select
    Call parent.ReplaceItemValue("CurrentOrder", Cstr(num + 1))
    Call parent.ComputeWithForm (True, False)
    Call parent.save (True, True)
    Print "[/"+db.filename+"/All+Documents/"+ _
                            doc.ParentDocumentUNID +"?OpenDocument]"
End Sub
```

## B.3 Complete Order Agent

```
(Option)
Option Public
Use "OrderCurrency"

Sub Initialize
    Dim  session As New  NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim dc As NotesDocumentCollection
    Dim docview As NotesDocument

    Set db = session.CurrentDatabase
    Set dc = db.AllDocuments
    Set doc = session.DocumentContext

    Call doc.Save (True, True)
    If doc.IsResponse Then Goto DuplicateErrorMessage

    Dim CurrOrder As String
    Dim CustomerOrder As String
    Dim CurrDenomi As String

    CurrOrder = CurrOrder +  _
    Left(doc.GetItemValue("OrderCurrency1")(0) + Space(207), 207) +_
    Left(doc.GetItemValue("OrderCurrency2")(0) + Space(207), 207) +_
    Left(doc.GetItemValue("OrderCurrency3")(0) + Space(207), 207) +_
    Left(doc.GetItemValue("OrderCurrency4")(0) + Space(207), 207) +_
    Left(doc.GetItemValue("OrderCurrency5")(0) + Space(207), 207)
    CurrDenomi = CurrDenomi + _
    Left(doc.GetItemValue("OrderDenomiQty1")(0) + Space(210),210) +_
    Left(doc.GetItemValue("OrderDenomiQty2")(0) + Space(210), 210) +_
```

```
     Left(doc.GetItemValue("OrderDenomiQty3")(0) + Space(210), 210) +_
     Left(doc.GetItemValue("OrderDenomiQty4")(0) + Space(210), 210) +_
     Left(doc.GetItemValue("OrderDenomiQty5")(0) + Space(210), 210)

     Call UpdateStockOrder (CurrOrder, CurrDenomi, CustomerOrder)
     Call doc.ReplaceItemValue ("OrderStatus", "Processed")
     Call doc.ReplaceItemValue ("CurrentOrder", "0")
     Call doc.Save (True, True)
     Exit Sub

DuplicateErrorMessage:
     Print "Duplicate Order"
End Sub
```

## B.4 Complete Order Offline

```
(Option)
Option Public
Use "OrderCurrency"

Sub Initialize
     Dim  session As New  NotesSession
     Dim db As NotesDatabase
     Dim doc As NotesDocument
     Dim dc As NotesDocumentCollection
     Dim Account As String
     Dim count As Integer
     Dim CurrOrder As String
     Dim CustomerOrder As String
     Dim CurrDenomi As String

     Set db = session.CurrentDatabase
     Set dc = db.AllDocuments
     For count = 1 To dc.count
          Set doc = dc.GetNthDocument(count)
          If doc.GetItemValue ("OrderStatus")(0) = "Request Process"  Then
               '*****************************
               ' *Get Branch Data Section
               '*****************************
               Call GetBranchData()
               Call doc.ReplaceItemValue ("BranchName", BRANCH_DATA)
               Call doc.ReplaceItemValue ("BranchCommRate",_
                                          Cstr(BRANCH_RATE))

               '*****************************
               ' *Get Cashier ID Section
               '*****************************
               If GetCashierID(doc.GetItemValue("CashierName")(0))=False _Then
                    Print "Contact System Administrator!!"
                    Exit Sub
               End If
               If CASHIER_ID = "" Then
                    Exit Sub
               Else
                    Call doc.ReplaceItemValue ("CashierID", CASHIER_ID)
                    Call doc.ReplaceItemValue ("CashierName",CASHIER_NAME)
                    Call doc.ReplaceItemValue ("CashierData", CASHIER_ID +_
                                               "  "  + CASHIER_NAME)
               End If

               '*****************************
```

```
' *Get OrderNumber  Section
'*****************************
Call GetNewOdrRef()
Call doc.ReplaceItemValue ("OrderNumber", ORDER_NUMBER)

'*****************************************************
' *Get Customer Information If customer number exist
'*****************************************************
account = doc.GetItemValue("AccountNum")(0)
If account <> "" Then
     Call  VerifyCustomerAccount (account)
     Call doc.ReplaceItemValue ("CustomerName",_
                                CUSTOMER_NAME)
     Call doc.ReplaceItemValue ("CustomerAddr",CUSTOMER_ADDR)
End If

'*****************************************************
' *Get Customer Information If customer number exist
'*****************************************************
doc.Form = "Order Currency"
Call doc.ComputeWithForm( False, False )
Call doc.Save (True, True)

'***********************************
' * Re Caluculation
'***********************************
If doc.GetItemValue ("OrderType1")(0) <> "" Then
     Call GetCurrDenomiList( _
                Left(doc.GetItemValue("OrderType1")(0),3))
     Call  doc.ReplaceItemValue ("CurrencyBuy", EXCHANGE_BUY)
     Call  doc.ReplaceItemValue ("CurrencySell",_
                                 EXCHANGE_SELL)
     Call doc.ComputeWithForm( False, False )
     OrderCurr1 = doc.GetItemValue("CashierID" )(0) +_
     Left(doc.GetItemValue ("OrderType1")(0),3) _
     + Space(20) +_ Right(Space(12) +_
     doc.GetItemValue("OrderTotal1")(0),12) + Space(172)
End If
If doc.GetItemValue ("OrderType2")(0) <> "" Then
     Call  GetCurrDenomiList(_
              Left(doc.GetItemValue("OrderType2")(0),3))
     Call  doc.ReplaceItemValue ("CurrencyBuy", EXCHANGE_BUY)
     Call  doc.ReplaceItemValue ("CurrencySell",_
                                 EXCHANGE_SELL)
     Call doc.ComputeWithForm( False, False )
     OrderCurr2 = doc.GetItemValue("CashierID" )(0) +_
     Left(doc.GetItemValue ("OrderType2")(0),3) _
     + Space(20) +_ Right(Space(12) +_
     doc.GetItemValue("OrderTotal2")(0),12) + Space(172)
End If
If doc.GetItemValue ("OrderType3")(0) <> "" Then
     Call  GetCurrDenomiList(_
           Left(doc.GetItemValue("OrderType3")(0),3))
     Call  doc.ReplaceItemValue ("CurrencyBuy", EXCHANGE_BUY)
     Call  doc.ReplaceItemValue ("CurrencySell",_
                                 EXCHANGE_SELL)
     Call doc.ComputeWithForm( False, False )
     OrderCurr3 = doc.GetItemValue("CashierID" )(0) +_
     Left(doc.GetItemValue ("OrderType3")(0),3) _
     + Space(20) +_ Right(Space(12) +_
     doc.GetItemValue("OrderTotal3")(0),12) + Space(172)
End If
If doc.GetItemValue ("OrderType4")(0) <> "" Then
```

```
                    Call  GetCurrDenomiList(_
                          Left(doc.GetItemValue("OrderType4")(0),3))
                    Call  doc.ReplaceItemValue ("CurrencyBuy", EXCHANGE_BUY)
                    Call  doc.ReplaceItemValue ("CurrencySell",_
                                          EXCHANGE_SELL)
                    Call doc.ComputeWithForm( False, False )
                    OrderCurr4 = oc.GetItemValue("CashierID" )(0) +_
                    Left(doc.GetItemValue ("OrderType4")(0),3) _
                    + Space(20) +_ Right(Space(12) +_
                    doc.GetItemValue("OrderTotal4")(0),12) + Space(172)
               End If
               If doc.GetItemValue ("OrderType5")(0) <> "" Then
                    Call  GetCurrDenomiList(_
                          Left(doc.GetItemValue("OrderType5")(0),3))
                    Call  doc.ReplaceItemValue ("CurrencyBuy", EXCHANGE_BUY)
                    Call  doc.ReplaceItemValue ("CurrencySell",_
                                          EXCHANGE_SELL)
                    Call doc.ComputeWithForm( False, False )
                    OrderCurr5 = doc.GetItemValue("CashierID" )(0) +_
                    Left(doc.GetItemValue ("OrderType5")(0),3) _
                    + Space(20) +_ Right(Space(12) +_
                    doc.GetItemValue("OrderTotal5")(0),12) + Space(172)
               End If

               '************************************
               ' * Complete Order
               '************************************
               CurrOrder = CurrOrder + Left(OrderCurr1 +Space(207), 207) _
                           +  Left(OrderCurr2 + Space(207), 207) _
                           +  Left(OrderCurr3 + Space(207), 207) _
                           +  Left(OrderCurr4 + Space(207), 207)_
                           +  Left(OrderCurr5 + Space(207), 207)
               CurrDenomi = CurrDenomi +_
               Left(doc.GetItemValue("OrderDenomiQty1")(0) _
               + Space(210), 210) +_
               Left(doc.GetItemValue("OrderDenomiQty2")(0) _
               + Space(210), 210) +_
               Left(doc.GetItemValue("OrderDenomiQty3")(0) _
               + Space(210), 210) +_
               Left(doc.GetItemValue("OrderDenomiQty4")(0) _
               + Space(210), 210)  +_
               Left(doc.GetItemValue("OrderDenomiQty5")(0) _
               + Space(210), 210)

               Call doc.ReplaceItemValue ("OrderStatus", "Processed")
               Call doc.ReplaceItemValue ("CurrentOrder", "0")
               Call doc.Save (True, True)
          End If
     Next
End Sub
```

## B.5 Garbage Response Agent

```
Sub Initialize
     Dim  session As New  NotesSession
     Dim db As NotesDatabase
     Dim doc As NotesDocument
     Dim dc As NotesDocumentCollection

     Set db = session.CurrentDatabase
     Set dc = db.AllDocuments
```

```
     For count = 1 To dc.count
          Set doc = dc.GetNthDocument(count)
          If doc.IsResponse  Then
               doc.Remove(True)
          End If
     Next
End Sub
```

## B.6 Get Customer Info Agent

```
(Option)
Option Public
Use "OrderCurrency"

Sub Initialize
     Dim  session As New  NotesSession
     Dim db As NotesDatabase
     Dim doc As NotesDocument
     Dim account As String
     Dim customer As String

     Set db = session.CurrentDatabase
     Set doc = session.DocumentContext

'*******************************************************
' *Get Customer Information If customer number exist
'*******************************************************
account = doc.GetItemValue("AccountNum")(0)

     If account <> "" Then
          Call  VerifyCustomerAccount (account)
          Call doc.ReplaceItemValue ("CustomerName", CUSTOMER_NAME)
          Call doc.ReplaceItemValue ("CustomerAddr",CUSTOMER_ADDR)
     End If

'*******************************************************
' *Reload Order Page
'*******************************************************
     Call doc.Save (True, True)
     Print "[/"+db.filename+"/All+Documents/"+ doc.NoteID +"?OpenDocument]"
End Sub
```

## B.7 Get Denomination Info Agent

```
(Option)
Option Public
Use "OrderCurrency"

Sub Initialize
     Dim  session As New  NotesSession
     Dim db As NotesDatabase
     Dim doc As NotesDocument
     Dim currtype As String

     Set db = session.CurrentDatabase
     Set doc = session.DocumentContext

'*************************************************
' *Get Selected Currency Type
```

```
'************************************************
     currtype = Left(doc.GetItemValue("CurrencyListN")(0),3)

     Call  GetCurrDenomiList (currtype)

     For count = 1 To MAX_DENOMI
          denomilist = denomilist + DENOMI(count) + ";"
          availlist = availlist + DSTOCK(count) + ";"
     Next
     Call doc.ReplaceItemValue("DenomiList", denomilist)
     Call doc.ReplaceItemValue("AvailList", availlist)

     Call  doc.ReplaceItemValue ("CurrencyBuy", EXCHANGE_BUY)
     Call  doc.ReplaceItemValue ("CurrencySell", EXCHANGE_SELL)
     Call doc.Save (True, True)
End Sub
```

## B.8 Get Order Number Agent

```
(Option)
Option Public
Use "OrderCurrency"

Sub Initialize
     Dim  session As New  NotesSession
     Dim db As NotesDatabase
     Dim doc As NotesDocument
     Dim account As String
     Dim customer As String
     Set db = session.CurrentDatabase
     Set doc = session.DocumentContext

'*****************************************
' *Get OrderNumber  Section
'*****************************************
     Call GetNewOdrRef()
     Call doc.ReplaceItemValue ("OrderNumber", ORDER_NUMBER)

'***********************************************
' *Reload Order Page
'***********************************************
     Call doc.Save (True, True)
     Print "[/"+db.filename+"/All+Documents/"+ doc.NoteID +"?OpenDocument]"
End Sub
```

# Appendix C. Action Buttons

In this appendix we list the code used for the action buttons.

## C.1 Get Order Reference Button

```
Sub Click(Source As Button)
     Dim uiWorkspace As New NotesUIWorkspace
     Dim uidoc As NotesUiDocument
     Dim doc As NotesDocument

     Set uidoc = uiWorkspace.CurrentDocument
     Set doc = uidoc.document
     Call GetNewOdrRef()
     Call uidoc.FieldSetText("OrderNumber", ORDER_NUMBER)
     Call uidoc.Save
     Call uidoc.Refresh
End Sub
```

## C.2 Verify Account Number Button

```
Sub Click(Source As Button)
     Dim uiWorkspace As New NotesUIWorkspace
     Dim uidoc As NotesUiDocument

     Set uidoc = uiWorkspace.CurrentDocument
     If  uidoc.FieldGetText("AccountNum") <> "" Then
          Call  VerifyCustomerAccount (uidoc.FieldGetText("AccountNum"))
     End If
     Call uidoc.FieldSetText("CustomerName", CUSTOMER_NAME)
     Call uidoc.FieldSetText("CustomerAddr",CUSTOMER_ADDR)
End Sub
```

## C.3 Add New Currency Button

```
Sub Click(Source As Button)
     Dim uiWorkspace As New NotesUIWorkspace
     Dim uidoc As NotesUiDocument
     Dim doc As NotesDocument
     Dim num As Integer
     Dim count As Integer
     Dim currorder As String
     Dim total  As Currency
     Dim denomiorder As String
     Dim number As String
     Dim denomilist As String
     Dim availlist As String
     Dim eval As Variant

     Set uidoc = uiWorkspace.CurrentDocument
     Set doc = uidoc.document

     If Trim(uidoc.FieldGetText("CurrencyListN")) = "" Then
          Messagebox "Please select a currency!!"
          Exit Sub
```

```
Else
     Call uidoc.FieldSetText("CurrencyType",_
                              Left(uidoc.FieldGetText("CurrencyListN"),3))
End If

If uidoc.FieldGetText ("CurrentOrder") <> MAX_ORDER Then
     Call  GetCurrDenomiList (uidoc.FieldGetText("CurrencyType"))

     For count = 1 To MAX_DENOMI
          denomilist = denomilist + DENOMI(count) + ";"
          availlist = availlist + DSTOCK(count) + ";"
     Next
     Call uidoc.FieldSetText("DenomiList", denomilist)
     Call uidoc.FieldSetText("AvailList", availlist)

     Call  uidoc.FieldSetText ("CurrencyBuy", EXCHANGE_BUY)
     Call  uidoc.FieldSetText ("CurrencySell", EXCHANGE_SELL)
     If uiWorkspace.Dialogbox("Currency Details Notes", True, True, False, True,_
                    False, False, "Currency Specification Facility") = True Then

          DORDER (1) = uidoc.FieldGetText("Order1")
          DORDER (2) = uidoc.FieldGetText("Order2")
          DORDER (3) = uidoc.FieldGetText("Order3")
          DORDER (4) = uidoc.FieldGetText("Order4")
          DORDER (5) = uidoc.FieldGetText("Order5")
          DORDER (6) = uidoc.FieldGetText("Order6")
          DORDER (7) = uidoc.FieldGetText("Order7")
          DORDER (8) = uidoc.FieldGetText("Order8")
          DORDER (9) = uidoc.FieldGetText("Order9")
          DORDER (10) = uidoc.FieldGetText("Order10")

          For count = 1 To MAX_DENOMI
               denomiorder = denomiorder + uidoc.FieldGetText ("CurrencyType") +_
             Right(Space(9) + DENOMI(count), 9) + Right(Space(9) + DORDER(count), 9)
          Next
          currorder = uidoc.FieldGetText("CashierID" ) + _
                      uidoc.FieldGetText ("CurrencyType") +_
                      Space(20) + Right(Space(12) +_
                      uidoc.FieldGetText("OrderTotal"),12) + Space(172)

          num = Cint(uidoc.FieldGetText("CurrentOrder"))
          Select  Case num
          Case  1:
               Call uidoc.FieldSetText ("OrderType1",_
                    uidoc.FieldGetText("CurrencyListN"))
               Call uidoc.FieldSetText ("OrderTotal1",_
                    uidoc.FieldGetText("OrderTotal"))
               Call uidoc.FieldSetText ("OrderCurrency1", currorder)
               Call uidoc.FieldSetText ("OrderDenomiQty1", denomiorder)
          Case 2:
               Call uidoc.FieldSetText ("OrderType2",_
                    uidoc.FieldGetText("CurrencyListN"))
               Call uidoc.FieldSetText ("OrderTotal2",_
                    uidoc.FieldGetText("OrderTotal"))
               Call uidoc.FieldSetText ("OrderCurrency2", currorder)
               Call uidoc.FieldSetText ("OrderDenomiQty2", denomiorder)
          Case 3:
               Call uidoc.FieldSetText ("OrderType3",_
                    uidoc.FieldGetText("CurrencyListN"))
               Call uidoc.FieldSetText ("OrderTotal3",_
                    uidoc.FieldGetText("OrderTotal"))
               Call uidoc.FieldSetText ("OrderCurrency3", currorder)
               Call uidoc.FieldSetText ("OrderDenomiQty3", denomiorder)
```

```
                Case 4:
                     Call uidoc.FieldSetText ("OrderType4",_
                          uidoc.FieldGetText("CurrencyListN"))
                     Call uidoc.FieldSetText ("OrderTotal4",_
                          uidoc.FieldGetText("OrderTotal"))
                     Call uidoc.FieldSetText ("OrderCurrency4", curorder)
                     Call uidoc.FieldSetText ("OrderDenomiQty4", denomiorder)
                Case 5:
                     Call uidoc.FieldSetText ("OrderType5",_
                          uidoc.FieldGetText("CurrencyListN"))
                     Call uidoc.FieldSetText ("OrderTotal5",_
                          uidoc.FieldGetText("OrderTotal"))
                     Call uidoc.FieldSetText ("OrderCurrency5", curorder)
                     Call uidoc.FieldSetText ("OrderDenomiQty5", denomiorder)
                Case Else
                     Messagebox "No more Order in this form!!"
                     Exit Sub
                End Select

                Call uidoc.FieldSetText ("LocalCommission",_
                     uidoc.FieldGetText("BranchCommRate"))
                Call uidoc.FieldSetText ("CurrencyDenomi", "")
                Call uidoc.FieldSetText ("OrderTotal", "")
                Call uidoc.FieldSetText ("OrderLocal", "")
                Call uidoc.FieldSetText ("OrderQty", "")
                Call uidoc.FieldSetText ("CurrentOrder", _
                     Cstr(Cint(uidoc.FieldGetText("CurrentOrder")) + 1))

                Call uidoc.FieldSetText("Order1","0")
                Call uidoc.FieldSetText("Order2","0")
                Call uidoc.FieldSetText("Order3","0")
                Call uidoc.FieldSetText("Order4","0")
                Call uidoc.FieldSetText("Order5","0")
                Call uidoc.FieldSetText("Order7","0")
                Call uidoc.FieldSetText("Order8","0")
                Call uidoc.FieldSetText("Order9","0")
                Call uidoc.FieldSetText("Order10","0")

           End If
      End If
      uidoc.Refresh
End Sub
```

## C.4 Complete Order Button

```
Sub Click(Source As Button)
     Dim uiWorkspace As New NotesUIWorkspace
     Dim uidoc As NotesUiDocument
     Set uidoc = uiWorkspace.CurrentDocument
     Dim CurrOrder As String
     Dim CustomerOrder As String
     Dim CurrDenomi As String
     CurrOrder = CurrOrder +Left(uidoc.FieldGetText("OrderCurrency1")+Space(207),207) +_
                 Left(uidoc.FieldGetText("OrderCurrency2") + Space(207), 207) +_
                 Left(uidoc.FieldGetText("OrderCurrency3") + Space(207), 207) +_
                 Left(uidoc.FieldGetText("OrderCurrency4") + Space(207), 207) +_
                 Left(uidoc.FieldGetText("OrderCurrency5") + Space(207), 207)
    CurrDenomi = CurrDenomi+Left(uidoc.FieldGetText("OrderDenomiQty1")+Space(210),210)+_
                 Left(uidoc.FieldGetText("OrderDenomiQty2") + Space(210), 210) +_
                 Left(uidoc.FieldGetText("OrderDenomiQty3") + Space(210), 210) +_
                 Left(uidoc.FieldGetText("OrderDenomiQty4") + Space(210), 210) +_
```

```
                Left(uidoc.FieldGetText("OrderDenomiQty5") + Space(210), 210)
     Call UpdateStockOrder (CurrOrder, CurrDenomi, CustomerOrder)
     Call uidoc.FieldSetText ("OrderStatus", "Processed")
     Call uidoc.FieldSetText ("CurrentOrder", "0")
     Call uidoc.Save
End Sub
```

## Appendix D. CICSCLI.INI

In this appendix we list the CICSCLI.INI file used to connect our Domino server to our CICS enterprise server.

```
;*********************************************************************
;* IBM CICS Client - Initialization File                             *
;*********************************************************************
;---------------------------------------------------------------------
; Client section - This section defines the local CICS client.  There
;                  should only be one Client section.

Client = *                      ; Auto-install client on the server
   MaxServers = 2               ; Only allow one server connection
   MaxRequests = 20             ; Limit the maximum server interaction
   MaxBufferSize = 32           ; Allow for a 32K maximum COMMAREA
   LogFile = CICSCLI.LOG        ; Set the error log file name
   TraceFile = CICSCLI.TRC      ; Set the trace log file name
   DumpFile = CICSCLI.DMP       ; Set the memory trace dump file name
   DumpMemSize = 16             ; Allow for 16k of trace in memory
   DosMemory = 48               ; The DOS client's memory pool size

;---------------------------------------------------------------------
; Server section - This section defines a server to which the client may
;                  connect.  There may be several Server sections.

Server = TCPWS                     ; Arbitrary name for the server
   Description = TCP/IP Server     ; Arbitrary description for the server
   Protocol = TCPIP                ; Matches with a Driver section below
   NetName =your.server.ip.address ; The server's TCP/IP address
   Port = 1435                     ; Use the default TCP/IP CICS port
   UpperCaseSecurity = N           ; Fold Userid and Password to uppercase
   InitialTransid = CLOG           ; Initial terminal transaction name

;---------------------------------------------------------------------
; Driver section - This section defines a communications protocol DLL
;                  used to communicate with a server.  There may be
;                  several Driver sections.

Driver = TCPIP                  ; Matches the Server's Protocol value
   DriverName = CCLWNTIP        ; Use the WinNT TCP/IP communications DLL

;*********************************************************************
;* End of file                                                       *
;*********************************************************************
```

## Appendix E. Special Notices

This publication is intended to help professionals use Domino to migrate client/server applications to network computing applications. The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Domino Version 4.6, MQSeries & CICS Connections for Domino 1.0, Lotus NotesPump Release 2.5. See the PUBLICATIONS section of the IBM Programming Announcement for each of those products 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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:

| | |
|---|---|
| CICS | CICS OS/2 |
| CICS/ESA | DB2 |
| DPROPR | IBM |
| MQ | MQSeries |
| MVS | MVS/ESA |
| OS/2 | System/390 |

The following terms are trademarks of the Lotus Development Corporation in the United States and/or other countries:

| | |
|---|---|
| 1-2-3 | Approach |
| Domino | Freelance |
| Lotus | Lotus Notes |
| LotusScript | Notes |
| NotesPump | NotesSQL |
| SmartSuite | WordPro |

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

# Appendix F. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## F.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 145.

Another ITSO publication covers the migration of the same client/server application to a Java model:

- *From Client/Server to Network Computing: A Migration to Java,* SG24-2247

Covering the integration of Lotus Notes and Domino applications with enterprise data and applications, the Lotus Solution for the Enterprise Collection contains:

- Volume 1 - *Lotus Notes: An Enterprise Application Platform*, SG24-4837
- Volume 2 - *Using DB2 in a Domino Environment*, SG24-4918 (to be published in 1998)
- Volume 3 - *Using the IBM CICS Gateway for Lotus Notes*, SG24-4512
- Volume 4 - *Lotus Notes and the MQSeries Enterprise Integrator*, SG24-2217
- Volume 5 - *NotesPump: The Enterprise Data Mover*, SG24-5255 (to be published in 1998)

These publications are also relevant as further information sources:

- *Enterprise Integration with Domino.Connect,* SG24-2181
- *Lotus Notes Release 4.5: A Developer's Handbook*, SG24-4876
- *The Domino Defense: Security in Lotus Notes and the Internet*, SG24-4848
- *Java Network Security, SG24-*2109
- *The Universal Connectivity Guide to DB2,* SG24-4894

## F.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

| CD-ROM Title | Subscription Number | Collection Kit Number |
|---|---|---|
| System/390 Redbooks Collection | SBOF-7201 | SK2T-2177 |
| Networking and Systems Management Redbooks Collection | SBOF-7370 | SK2T-6022 |
| Transaction Processing and Data Management Redbook | SBOF-7240 | SK2T-8038 |
| Lotus Redbooks Collection | SBOF-6899 | SK2T-8039 |
| Tivoli Redbooks Collection | SBOF-6898 | SK2T-8044 |
| AS/400 Redbooks Collection | SBOF-7270 | SK2T-2849 |
| RS/6000 Redbooks Collection (HTML, BkMgr) | SBOF-7230 | SK2T-8040 |
| RS/6000 Redbooks Collection (PostScript) | SBOF-7205 | SK2T-8041 |
| RS/6000 Redbooks Collection (PDF Format) | SBOF-8700 | SK2T-8043 |
| Application Development Redbooks Collection | SBOF-7290 | SK2T-8037 |

## F.3 Other Publications and Web Sites

These publications and Web sites are also relevant as further information sources:

### F.3.1 Network Computing Framework

- Web Sites
  - *http://www.software.ibm.com/openblue*, for information about IBM Open Blueprint
  - *http://www.software.ibm.com/ebusiness/ncf*, for information about the Network Computing Framework

### F.3.2 Domino

- Publications

  The Application Developer Doc Pack (Lotus SKU 027645) contains the following publications:

  - *Lotus Notes Designer for Domino 4.6 Application Developer's Guide*
  - *Lotus Notes Designer for Domino 4.6 Programmer's Guide*
  - *Lotus Notes Designer for Domino 4.6 Release Notes*
  - *Lotus Notes Release 4.5 Database Manager's Guide*

- *LotusScript 4.6 Classes Poster*
- *Application Development Roadmap*
- *Lotus Notes 4.6 Release Notes*
- *Lotus Domino 4.6 Release Notes*
- *LotusScript 3.1 Language Reference Manual*
- *LotusScript Programmer's Guide*

- Web Sites
  - *http://www.lotus.com*, for information about Lotus
  - *http://domino.lotus.com*, for information about Domino
  - *http://www.eicentral.lotus.com*, for information about Lotus enterprise integration

### F.3.3 CICS

- Publications
  - *CICS Transaction Server for OS/390 V1R2 Planning for Installation*, GC33-1789
  - *CICS Transaction Server for OS/390 V1R2 Release Guide*, GC33-1570
  - *CICS Transaction Server for OS/390 V1R2 Migration Guide*, GC33-1571
- Web Sites
  - *http://www.software.ibm.com/ts/cics*, for information about CICS

### F.3.4 DB2

- Publications
  - *IBM DB2 Universal Database Administration Getting Started*, S10J-8154
  - *IBM DB2 Universal Database Administration Guide Version 5*, S10J-8157
  - *IBM DB2 Universal Database Messages Reference Version 5*, S10J-8168
  - *IBM DB2 Universal Database Roadmap to DB2 Programming*, S10J-8155
  - *IBM DB2 Universal Database SQL Reference Version 5*, S10J-8165
- Web Sites
  - *http://www.software.ibm.com/data/db2/udb*, for information about DB2 Universal Database Version 5

# How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com`.

## How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** – to order hardcopies in United States
- **GOPHER link to the Internet** – type `GOPHER WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

## How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) – send orders to:

| | **IBMMAIL** | **Internet** |
|---|---|---|
| In United States | usib6fpl at ibmmail | usib6fpl@ibmmail.com |
| In Canada | caibmbkz at ibmmail | lmannix@vnet.ibm.com |
| Outside North America | dkibmbsh at ibmmail | bookshop@dk.ibm.com |

- **Telephone orders**

| | |
|---|---|
| United States (toll free) | 1-800-879-2755 |
| Canada (toll free) | 1-800-IBM-4YOU |
| Outside North America | (long distance charges apply) |
| (+45) 4810-1320 - Danish | (+45) 4810-1020 - German |
| (+45) 4810-1420 - Dutch | (+45) 4810-1620 - Italian |
| (+45) 4810-1540 - English | (+45) 4810-1270 - Norwegian |
| (+45) 4810-1670 - Finnish | (+45) 4810-1120 - Spanish |
| (+45) 4810-1220 - French | (+45) 4810-1170 - Swedish |

- **Mail Orders** – send orders to:

| | | |
|---|---|---|
| IBM Publications | IBM Publications | IBM Direct Services |
| Publications Customer Support | 144-4th Avenue, S.W. | Sortemosevej 21 |
| P.O. Box 29570 | Calgary, Alberta T2P 3N5 | DK-3450 Allerød |
| Raleigh, NC 27626-0570 | Canada | Denmark |
| USA | | |

- **Fax** – send orders to:

| | |
|---|---|
| United States (toll free) | 1-800-445-9269 |
| Canada | 1-800-267-4455 |
| Outside North America | (+45) 48 14 2207 (long distance charge) |

- **1-800-IBM-4FAX (United States)** or **(+1) 408 256 5422 (Outside USA)** – ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** – send note to `softwareshop@vnet.ibm.com`

- **On the World Wide Web**

| | |
|---|---|
| Redbooks Web Site | http://www.redbooks.ibm.com |
| IBM Direct Publications Catalog | http://www.elink.ibmlink.ibm.com/pbl/pbl |

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank).

## IBM Redbook 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

**@Dbfunction.** A fast and easy-to-use read-only access method to Lotus Notes or ODBC-compliant databases

**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 perform routine Lotus Notes tasks in the background. You can program agents, using LotusScript or the Lotus Notes formula language.

**applet.** A Java program designed to run within a Web browser. Contrast with application.

**application.** In Java programming, a self-contained, stand-alone Java program that includes a main() method. Contrast with applet.

**asynchronous.** Without regular time relationship; unexpected or unpredictable with respect to the execution of a program instruction. See *synchronous*.

**browser**. An Internet-based tool that lets users browse Web sites.

**business process.** An entity-handling activity that is of limited duration, defined in scope, and set by business goals and policies, not by organization or implementation.

**call level interface (CLI)**. A callable application programming interface (API) for database access, which is an alternative to an embedded SQL application program interface. 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 run time.

**Customer Information Control System (CICS).** A distributed online transaction processing system designed to support a network of many terminals. The CICS family of products is available for a variety of platforms ranging from a single workstation to the largest mainframe.

**CICS Client**. A server program that processes CICS ECI calls, forwarding transaction requests to a CICS program running on a host.

**class**. An aggregate that defines properties, operations, and behavior for all instances of that aggregate.

**client.** As in client/server computing, the application that makes requests to the server and often handles the necessary interaction with the user.

**client/server.** 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* and *server*.

**commit**. The operation that ends a unit of work to make permanent the changes it has made to resources (transaction or data).

**Common Gateway Interface (CGI)**. A standard protocol through which a Web server can execute programs running on the server machine. CGI programs are executed in response to requests from Web client browsers.

**communications area (COMMAREA)**. In a CICS transaction program, a group of records that describes both the format and volume of data used.

**conversational.** A communication model where two distributed applications exchange information by way of a conversation; typically one application starts (or allocates) the conversation, sends some data, and allows the other application to send some data. Both applications continue in turn until one decides to finish (or deallocate). The conversational model is a synchronous form of communication.

**database.** (1) A collection of related 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 Domino and Lotus Notes, a group of documents and their forms, views, folders, shared fields, subforms, navigators, script libraries, and agents, stored under one file.

**database management system (DBMS).** A computer program that manages data by providing the services of centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency control, privacy, and security.

**DB2 Call Level Interface (CLI).** The DB2 call level interface is an alternative SQL interface for the DB2 family of products and takes full advantage of DB2 capability.This implementation closely follows industry standards, such as X/OPEN, to enhance application portability. Currently, the DB2 CLI functions are compatible with ODBC 2.0 and contain DB2-specific APIs to help exploit DB2 capability.

**DB2 for MVS/ESA**. An IBM relational database management system for the MVS operating system.

**DCE.** Distributed Computing Environment. Adopted by the computer industry as a de facto standard for distributed computing. DCE allows computers from a variety of vendors to communicate transparently and share resources such as computing power, files, printers, and other objects in the network.

**distributed processing.** Distributed processing is an application or systems model in which function and data can be distributed across multiple computing resources connected on a LAN or WAN.

**distributed program link (DPL).** Enables an application program executing in one CICS system to link (pass control) to a program in a different CICS system. The linked-to program executes and returns a result to the linking program. This process is equivalent to remote procedure calls (RPCs). You can write applications that issue RPCs that can be received by members of the CICS family.

**document.** In Lotus Notes, a document is an object containing text, graphics, video, or audio or any kind of rich text data.

**dynamic link library (DLL)**. A file containing executable code and data bound to a program at run time rather than at link time. The C++ Access Builder generates beans and C++ wrappers that let your Java programs access C++ DLLs.

**e-business** Either (a) the transaction of business over an electronic medium such as the Internet or (b) a business that uses Internet technologies and network computing in its internal business processes (through intranets), its business relationships (through extranets), and the buying and selling of goods, services, and information (through electronic commerce.)

**external call interface (ECI).** An API that enables a non-CICS client application to call a CICS program as a subroutine. The client application communicates with the server CICS program, using a data area called a *COMMAREA*.

**extranet**. A private, virtual network that uses access control and security features to restrict the use of one or more intranets attached to the Internet to selected subscribers (such as personnel from a sponsoring company and its business partners).

**formula.** A Lotus Notes programming language that contains a set of built-in macros, functions, and commands.

**file transfer protocol (FTP).** The basic Internet function that enables files to be transferred between computers. You can use it to download files from a remote host computer, as well as to upload files from your computer to a remote host computer.

**gateway**. A host computer that connects networks that communicate in different languages. For example, a gateway connects a company's LAN to the Internet.

**graphical user interface (GUI)**. A type of interface that enables users to communicate with a program by manipulating graphical features rather than by entering commands. Typically, a

graphical user interface includes a combination of graphics, pointing devices, menu bars and other menus, overlapping windows, and icons.

**hypertext markup language (HTML).** The basic language that is used to build hypertext documents on the World Wide Web. It is used in basic, plain ASCII-text documents, but when those documents are interpreted (called *rendering*) by a Web browser such as Netscape, they can display formatted text, color, a variety of fonts, graphics images, special effects, hypertext jumps to other Internet locations, and information forms.

**hypertext transfer protocol (HTTP)**. The protocol for moving hypertext files across the Internet. Requires an HTTP client program on one end, and an HTTP server program on the other end. HTTP is the most important protocol used in the World Wide Web

**hypertext**. A way of presenting information online with connections (called *hypertext links*) between one piece of information and another.

**Internet Inter-ORB Protocol (IIOP)**. An industry standard protocol that defines how General Inter-ORB Protocol (GIOP) messages are exchanged over a TCP/IP network. The IIOP makes it possible to use the Internet itself as a backbone ORB through which other ORBs can bridge.

**integrated development environment (IDE)**. A software program comprising an editor, a compiler, and a debugger. IBM's VisualAge for Java is an example of an IDE.

**interface**. A set of methods that can be accessed by any class in the class hierarchy. The Interface page in the Workbench lists all interfaces in the workspace.

**Internet**. A collection of interconnected networks that use the Internet suite of protocols. The internet that allows universal access is referred to as the Internet (with a capital "I").

**Internet suite of protocols.** A set of protocols developed for use on the Internet and published as Requests for Comments (RFCs) through the Internet Engineering Task Force (IETF).

**intranet**. A private network that integrates Internet standards and applications (such as Web browsers) with an organization's existing computer networking infrastructure.

**Internet protocol (IP).** The rules that provide basic Internet functions.See *TCP/IP*.

**Java**. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

**JavaBeans**. The platform-independent, component architecture for the Java programming language. JavaBeans enables software developers to assemble pieces of Java code ("Beans") into a graphical drag-and-drop development environment.

**Java database connectivity (JDBC)**. An API that has the same characteristics as ODBC but is specifically designed for use by Java database applications. Also, for databases that do not have a JDBC driver, JDBC includes a JDBC to ODBC bridge. JDBC was developed by Sun Microsystems, Inc. and various partners and vendors.

**Java Development Kit (JDK)** The Java Development Kit 1.1 is the latest set of Java technologies made available to licensed developers by Sun Microsystems. Each release of the JDK contains the following: the Java Compiler, Java Virtual Machine, Java Class Libraries, Java Applet Viewer, Java Debugger, and other tools.

**JavaScript**. A scripting language that resembles Java and was developed by Netscape for use with the Netscape browser.

**local area network (LAN)**. A computer network located at a user's establishment within a limited geographical area. A LAN typically consists of one or more server machines providing services to a number of client workstations.

**logical unit of work (LUW).** An update that durably transforms a resource from one consistent state to another consistent state. 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 synch point on the system log), they do not need to be backed out after a subsequent error within the task or region. The end of an LUW is marked in a transaction by a synch point that is issued by either the user program or the CICS server, at the end of task. If there are no user synch points, the entire task is an LUW.

**LotusScript.** The Lotus cross-product BASIC scripting language.

**LU type 6.2 (LU 6.2).** A type of logical unit used for CICS intersystem communication (ISC). LU 6.2 architecture supports CICS host-to-system-level products and CICS host-to-device-level products. Advanced Program-to-Program Communication (APPC) is the protocol boundary of the LU 6.2 architecture.

**message.** In MQSeries, a string of bytes that one program wants to send to another.

**messaging.** A communications model whereby the distributed applications communicate by sending messages to each other. A message is typically a short packet of information that does not necessarily require a reply. Messaging implements asynchronous communications.

**method.** A fragment of code within a class that can be invoked and passed a set of parameters to perform a specific task.

**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 delivery of service requests or responses transparently across computing resources.

**MQSeries.** The MQSeries family of products provides APIs that allow you to use message queues to code indirect program-to-program communication.

**MQSeries Enterprise Integrator.** A LotusScript extension with special extensions and databases that allows the application developer to integrate Lotus Notes applications with MQSeries, IMS, and CICS applications.

**MQSeries LotusScript Extensions.** A LotusScript extension that allows the application developer to use the MQSeries APIs in a LotusScript application.

**Multipurpose Internet Mail Extension (MIME).** The Internet standard for mail that supports text, images, audio, and video.

**online transaction processing (OLTP).** A style of computing that supports interactive applications in which requests submitted by terminal users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. An OLTP system supervises the sharing of resources to allow efficient processing of multiple transactions at the same time.

**object**. (1) A computer representation of something that a user can work with to perform a task. An object can appear as text or an icon. (2) A collection of data and methods that operate on that data, which together represent a logical entity in the system. In object-oriented programming, objects are grouped into classes that share common data definitions and methods. Each object in the class is said to be an instance of the class. (3) An instance of an object class consisting of attributes, a data structure, and operational methods. It can represent a person, place, thing, event, or concept. Each instance has the same properties, attributes, and methods as other instances of the object class, though it has unique values assigned to its attributes.

**ODBC driver.** A dynamic link library (DLL) that implements ODBC function calls and interacts with a data source.

**ODBC driver manager.** A DLL, provided by Microsoft, with an import library. The primary purpose of the driver manager is to load ODBC drivers. The driver manager also provides entry points to ODBC functions for each driver and parameter validation and sequence validation for ODBC calls.

**open database connectivity (ODBC)**. A Microsoft-developed C database application programming interface (API) that allows access to database management systems calling callable SQL, which does not require the use of an SQL preprocessor. In addition, ODBC provides an architecture that allows users to add modules called *database drivers* that link the application to their choice of database management systems at run time. Thus applications no longer need to be directly linked to the modules of all of the database management systems that are supported.

**object request broker (ORB).** A CORBA term designating the means by which objects transparently make requests and receive responses from objects, whether they are local or remote.

**protocol**. (1) The set of all messages to which an object will respond. (2) Specification of the structure and meaning (the semantics) of messages that are exchanged between a client and a server. (3) Computer rules that provide uniform specifications so that computer hardware and operating systems can communicate. It is similar to the way that mail, in countries around the world, is addressed in the same basic format so that postal workers know where to find the recipient's address, the sender's return address, and the postage stamp. Regardless of the underlying language, the basic protocols remain the same.

**queue.** An MQSeries object. A queue is an area of storage set aside by the queue manager to hold messages on their way from one program to another.

**recovery.** The use of archived copies to reconstruct files, databases, or complete disk images after they are lost or destroyed.

**remote method invocation (RMI)**. In JDK 1.1, the API that allows you to write distributed Java programs, allowing methods of remote Java objects to be accessed from other Java virtual machines.

**remote procedure call (RPC)**. A communication model where requests are made by function calls to distributed procedures elsewhere. The location of the procedures is transparent to the calling application.

**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.

**server**. A computer that provides services to multiple users or workstations in a network; for example, a file server, a print server, or a mail server.

**Socket Secure (SOCKS).** The gateway that allows compliant client code (client code made socket secure) to establish a session with a remote host.

**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.

**synchronous.** (1) Pertaining to two or more processes that depend on the occurrence of a specific event such as a common timing signal. (2) Occurring with a regular or predictable time relationship.

**Transmission Control Protocol/Internet Protocol (TCP/IP).** The basic programming foundation that carries computer messages around the globe through the Internet. The suite of protocols that defines the Internet. Originally designed for the UNIX operating system, TCP/IP software is now available for every major kind of computer operating system. To be truly on the Internet, your computer must have TCP/IP software.

**thin client.** Usually refers to a system that runs on a resource-constrained machine or a small operating system. Thin clients do not require local system administration, and they execute Java applications delivered over the network.

**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.

**two-phase commit.** For a database, a protocol that is used to ensure uniform transaction commit or abort in a distributed data environment between two or more participants. The protocol consists of two phases: the first to reach a common decision, and the second to implement the decision.

**uniform resource locator (URL).** Standard to identify resources on the World Wide Web.

**unit of work.** A recoverable sequence of operations performed by an application between two points of consistency.

**virtual machine (VM)** A software program that executes other computer programs. It allows a physical machine, a computer, to behave as if it were another physical machine.

**Web browser.** A client program that initiates requests to a Web server and displays the information that the server returns.

**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.

**workstation.** A configuration of input and output equipment at which an operator works. A terminal or microcomputer, usually one that is connected to a mainframe or a network, at which a user can perform applications.

**World Wide Web (WWW or Web).** A graphic hypertextual multimedia Internet service.

## List of Abbreviations

| | |
|---|---|
| **API** | application programming interface |
| **ACL** | access control list |
| **ARM** | application request manager |
| **ASCII** | American National Standard Code for Information Interchange |
| **CAE** | client application enabler |
| **CGI** | Common Gateway Interface |
| **CICS** | Customer Information Control System |
| **CLI** | call level interface |
| **COMMAREA** | communication area (CICS) |
| **CORBA** | Common Object Request Broker Architecture |
| **DBMS** | database management system |
| **DB2LSX** | DB2 LotusScript Extension |
| **DCE** | distributed computing environment |
| **DDCS/2** | Distributed Database Connection Services/2 |
| **DLL** | dynamic link library |
| **DPL** | distributed program link (CICS) |
| **DRDA** | Distributed Relational Database Architecture |
| **EBCDIC** | extended binary coded decimal interchange code |
| **ECI** | external call interface (CICS) |
| **ERP** | enterprise resource planning |
| **ESA** | Enterprise Systems Architecture |
| **FTP** | file transfer protocol |
| **GUI** | graphical user interface |
| **HTML** | Hypertext Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **IBM** | International Business Machines Corporation |
| **IDE** | integrated development environment |
| **IIOP** | Internet Inter-ORB Protocol |
| **IMAP** | Internet Message Access Protocol |
| **ITSO** | International Technical Support Organization |
| **JDBC** | Java database connectivity |
| **JDK** | Java Development Kit |
| **JVM** | Java virtual machine |
| **LAN** | local area network |
| **LDAP** | Lightweight Directory Access Protocol |
| **LS:DO** | LotusScript Data Option |
| **LSX** | LotusScript Extension |
| **LUW** | logical unit of work |
| **MAPI** | messaging application program interface |
| **MIME** | Multipurpose Internet Mail Extension |

| | |
|---|---|
| **MQEI** | MQSeries Enterprise Integrator |
| **MQI** | message queue interface |
| **MQLSX** | MQSeries link LotusScript Extension |
| **MVS** | Multiple Virtual Storage |
| **MTA** | message transfer agent |
| **NC** | network computer |
| **NCF** | network computing framework |
| **NetBIOS** | Network Basic Input/Output System |
| **NNTP** | NetNews Transfer Protocol |
| **NT** | Microsoft Windows NT (new technology) |
| **OCI** | Oracle call interface |
| **ODBC** | open database connectivity |
| **OLTP** | online transaction processing |
| **ORB** | object request broker |
| **OS/2** | Operating System/2 |
| **OSF** | Open Software Foundation |
| **PC** | personal computer |
| **POP** | Post Office Protocol |
| **RACF** | Resource Access Control Facility |
| **RAD** | rapid application development |
| **RMI** | remote method invocation |
| **RSA** | Rivest Shamir Adleman |
| **SDK** | software developer's kit |
| **SET** | secure electronic transaction |
| **SMTP** | Simple Mail Transfer Protocol |
| **SNA** | Systems Network Architecture |
| **SNMP** | Simple Network Management Protocol |
| **SMTP** | Simple Mail Transfer Protocol |
| **SQL** | structured query language |
| **SSL** | secure sockets layer |
| **TCP/IP** | Transmission Control Protocol/Internet Protocol |
| **UPM** | User Profile Manager |
| **URL** | uniform resource locator |
| **XA** | extended architecture |

# Index

## Symbols



## Numerics



## A



## B



## C

## D

## E

## F



## G



## H



## I

## J



## K



## L

## M



## N

## O



## P



## R

## S



## T



## U

## V



## W



## X

# ITSO Redbook Evaluation

From Client/Server to Network Computing A Migration to Domino
SG24-5087-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ **Customer** _ **Business Partner** _ **Solution Developer** _ **IBM employee**
_ **None of the above**

**Please rate your overall satisfaction** with this book using the scale:
**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

Overall Satisfaction _____

**Please answer the following questions:**

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

**Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

SG24-5087-00
Printed in the U.S.A.