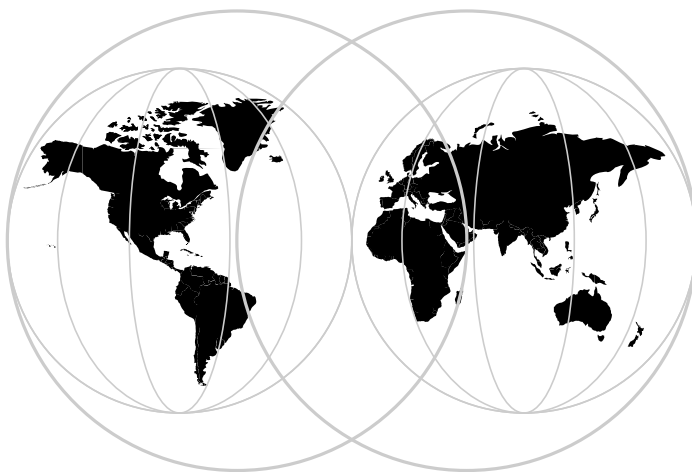




Lotus Notes Release 4.5: A Developer's Handbook



International Technical Support Organization

Edition Notice

First Edition (November 1996)

This edition applies to Release 4.5 of Lotus Notes.

Comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

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.

© International Business Machines Corporation 1996. All rights reserved.

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

Abstract

This document describes how to develop applications with Lotus Notes Release 4.5. Lotus Notes is a distributed client/server platform that allows you to develop applications to be shared by groups of users across a network. The Integrated Development Environment (IDE) provided by Notes enables the development of strategic enterprise-wide business applications.

Part 1 covers features such as databases, forms, subforms, views, action bars, layout regions, navigators, and agents. One chapter illustrates how to use LotusScript to write object-oriented sophisticated applications. Other chapters show you how to work with LotusScript Extensions, Lotus Components, and how to use the application development features of calendaring and scheduling. A workflow example is described in more detail. Examples and useful hints and tips are provided throughout.

Part 2 is dedicated to tools and techniques that enable you to seamlessly integrate an existing IT infrastructure into your Notes applications. Several chapters take a close look at Domino, the integrated Notes/HTTP server, and the programmability features provided by this environment. Following this, we discuss the integration options that exploit the respective strengths of Notes and traditional database management systems and transaction systems. A case study illustrates the various features throughout part 2 of the book. The last chapter discusses the Notes C++ API.

The book comes with a chart listing the LotusScript classes, methods, and properties based on Notes Release 4.5, and a diskette containing several sample applications.

Chapter 1

Getting Started



What Is Lotus Notes?

Lotus Notes® is a distributed client / server platform that allows you to develop applications containing data to be shared by groups of users across a network. It is comprised of a set of document databases that reside on top of a messaging infrastructure. Leveraging the distributed storage and messaging features, the Integrated Development Environment (IDE) provided by Notes™ enables Rapid Application Development and Deployment (RADD) of strategic enterprise-wide business applications.

Notes Is an Environment for Application Development Deployment

Lotus Notes Release 4.5 contains a powerful, distributed, document-oriented database that combines information storage with enterprise-wide communications, collaboration and coordination capabilities. Using the native set of application development tools, you can create business applications that store and route information objects. Notes contains a document-oriented, distributed database that opens a new world of sophisticated business applications based on the interchange of information, not just data.

Notes ADE Is Mainstream, Modern, and Industrial

Notes is comprised of a document database that is integrated into an enterprise messaging infrastructure. The integral Notes Application Development Environment (ADE) enables rapid development of strategic business applications using these database and messaging services.

The Notes ADE is fast and efficient, as well as easy to learn. Notes offers two compelling advantages for client / server application development:

- It is cross-platform, supporting a wide range of operating and network systems as well as hardware platforms.

- Notes applications are self-deploying. Replication is a key Notes function that synchronizes client and server databases. Since Notes can use the replication process to distribute design element changes, Notes uniquely offers the industry's only seamless client/server cross-platform deployment capability.

The Notes ADE offers developers architectural solutions to many of the challenges faced by client/server application development and deployment. Notes Release 4.5 achieves this using groundbreaking improvements in application development and deployment. Notes' ADE is:

- *Mainstream*

Notes applications will be used by everyone in your company since the applications are easy to create and maintain, and can be readily deployed across any network infrastructure.

Notes developers work in the same Notes environment as end users. This means that application design and development are iterative and can easily include end users in the design and development process. This reduces the time it takes to create Notes applications and ensures that the delivered application meets end users' needs.

- *Modern*

Notes represents a new generation of technology for the development of network-centric applications. The Notes ADE provides the latest development technology such as object-oriented tools and high-level APIs. LotusScript® is a structured programming language that provides a powerful programming environment in Notes. This allows for development of sophisticated and complex applications. Notes also contains capabilities that allow developers to integrate Notes with their existing relational database applications.

Notes is also modern because it permits effortless client/server deployment. Notes applications are isolated from platform and operating systems environments. Any application developed on one Notes client platform will run — without recompiles or switches — on any other Notes client or server platform. Now developers can create applications on UNIX for the Apple Macintosh, or on Windows 95 for OS/2.

- *Industrial*

Notes has been available since 1990. This is the fourth release, which incorporates the experiences and requirements of tens of thousands of developers over time. Notes is a tried and true application development platform, having successfully delivered compelling network applications to millions of end users.

The databases you develop can be dedicated to a specific group of users, to a department, or they can be a strategic, enterprise-wide application, providing both communication flow and an information warehouse.

From Notes Release 4.0 you now have support for databases up to four gigabytes (up from one gigabyte in Release 3.x) and many more users per server than before.

Note Each database can be up to four gigabytes. You can have multiple databases on the same server.

All the Notes Release 3 and Release 4 applications that you have developed will run unmodified in Release 4.5. Its messaging infrastructure uses your existing network. Your Notes development skills are preserved. The addition of LotusScript leverages widespread knowledge of the BASIC language.

Notes Is Complete

Notes has a wide set of robust built-in features that are focused on groupware application development:

Multi-platform

One of the key features of Notes is its ability to run on numerous platforms, both for the server and the client workstations. This is key to the ability to communicate not only within your own organization but also with your suppliers and customers: think of the flexibility you can achieve with Notes because the decisions about applications are now separated from the decisions about corporate infrastructure and operating systems.

The Notes Server can run on the following operating systems:

- Microsoft Windows NT
- Microsoft Windows 95
- IBM OS/2 Warp 4.0
- IBM OS/2 Warp 3.0 and OS/2 2.1
- UNIX: IBM AIX, Sun Solaris, SCO, HP/UX

The Notes Client can run on the following platforms:

- Microsoft Windows NT, Windows 95 and Windows 3.1 or 3.11
- IBM OS/2 Warp 4.0
- IBM OS/2 Warp 3.0 and OS/2 2.1
- Apple System 7.x
- UNIX: IBM AIX, Sun Solaris, SCO, HP/UX

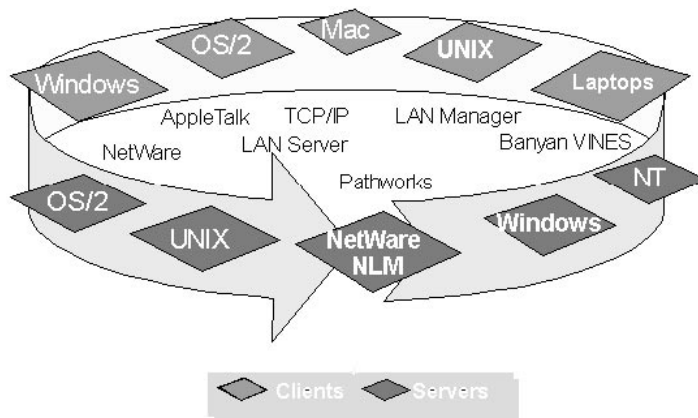
Notes supports the following Network Operating Systems (NOS):

- Novell NetWare
- IBM LAN Server
- Microsoft LAN Manager
- Microsoft NT Advanced Server
- PATHWORKS
- Banyan VINES
- AppleTalk

Notes supports the following Network Protocols:

- NetBIOS
- NetBEUI
- TCP/IP
- IPX and SPX
- AppleTalk
- Any combination of the above

The following figure gives an overview of the supported platforms and network protocols:



Not only can Notes run your applications on different operating systems, but the Notes application itself also handles all the cross-platform aspects that developers have to deal with. This means:

- You do not need to know different sets of operating system APIs
- You do not need to port the code from one platform to the other

- You do not need to recompile any source code
- You do not need to worry about communication aspects across heterogeneous networks

You have a wholly integrated deployment environment for completed applications, no matter the target operating system, network, server, or client.

The result is increased productivity, which is also supported by the following features:

- A common user interface
- The same development tools and language, both for the @Functions and LotusScript

Platform-specific features can, of course, be fully exploited.

Integrated Notes HTTP Server

Lotus Notes has a powerful, integrated HTTP server that allows you to create and manage Web sites directly from within the Notes environment. This allows you to leverage the existing power of the Notes ADE to publish documents and interact with customers over the Internet.

Lotus Notes comes with the Net.Action tool that allows you to set up and implement your first Web site with the minimal of effort, all from within Notes.

Integrated Messaging

Notes applications benefit from the integrated messaging infrastructure. They can route documents to selected users or groups of users on the Local Area Network (LAN) or Wide Area Network (WAN). Those documents can be used as notifications, reminders or requests for approvals.

Again, developers do not need to know the semantics of the underlying messaging protocols (VIM or MAPI) and can stay focused on the business solution.

Native Calendaring & Scheduling

New to Notes 4.5 is native calendaring and scheduling modeled on the Lotus Organizer user interface. In your mail file you now have the ability to book meetings with other Notes users, check their availability, reserve rooms and resources, create personal appointments and set alarms. With calendaring and scheduling comes a new programmable view based on a calendar-style appearance that can be added to any new or existing

database. This view can be displayed in one of four different formats: 2 days, 1 week, 2 weeks or 1 month. It can contain any information related to a date and time.

Wide Range of Effective Security Features

In addition to the security features available for server access, Notes provides developers with multiple levels of security that can be built into their Notes applications:

- At the server level, public and private key authentication ensures that only authorized users can access a server.
- At the database level, the database administrator can assign different access levels to users or groups of users in the Access Control List (ACL). The end user can also decide to encrypt the local replica of the database.
- At the workstation level, new to Release 4.5 are Execution Control Lists (ECLs). Now the administrator and user have full control over what level of access a Notes application has to their workstation's resources.
- At form and view level, the developer can restrict the use of a particular form or view.
- At document level, the owner of the document can restrict the access to the document to specific users or groups of users, both in read-only mode or in update mode.
- At field level, the developer can implement automatic encryption. The end user can also encrypt fields on a document basis.

Effective Support of Mobile Users

Notes provides mobile users with seamless access to the data of the databases stored on the Notes servers and clients. Whether they are at home, in a hotel or in a business location, mobile users use replica copies of Notes databases. They can connect to the server through modems and can reach and update the information in the same way as their colleagues who are connected to the network.

Mobile users can also work completely in standalone mode and then simply exchange data with the server at a time of day when the telecommunication costs are low.

Release 4.5 of Notes now gives mobile users the ability to create a minimal copy of the Public Name and Address book so mail can be addressed and encrypted correctly while away from the office.

Access to External Data

Notes allows for easy access to non-Notes data that are stored on the workstation, on the LAN or even on a mainframe if gateways are installed. You can access that data using a variety of means:

- Object Linking and Embedding (OLE), with OLE compliant applications such as Lotus Windows SmartSuite and Microsoft Office products.
- Notes/FX 2.0 builds on earlier versions of Notes/FX by allowing the bi-directional transfer of data, properties, and methods between Notes/FX 2.0-enabled OLE 2.0 applications and Notes Release 4.5 applications. For example, it is now possible to pass methods to Notes/FX 2.0 applications, which makes it easier to display the Notes action bar within the Notes/FX 2.0-enabled application. This also allows for a more seamless integration of desktop applications within a Notes Release 4.5 workflow application.
- @DBCommand, @DBLookUp and @DBCColumn functions to relational databases.
- Open Data Base Connectivity (ODBC) drivers to relational databases.
- LotusScript Data Object (LS:DO).
- LotusScript eXtensions (LSX) to relational and transaction databases.

Notes Is a Document Database

Notes is comprised of databases which contain documents. In Notes, a document is defined as an object containing text, graphics, video, or audio objects or any other kind of “rich text” data. Notes databases are semi-structured records consisting of the following basic design elements:

- *Forms*
Notes developers and users can create forms to provide a structure for information entry and storage in the document.
- *Subforms*
A subform is an object within a form that can be reused across applications. A logo is a simple example. Subforms can also be used to contain logic and processing that should apply across all portions of the Notes application.
- *Collapsible sections*
You can also create sections within a form that can be expanded or collapsed depending on the need to view that particular piece of information. For example, you may create a form that has an explanatory section which can be collapsed after it has been read.

- *Fields*

A field is that part of a form containing a single type of information. There are several types of Notes fields, including rich text, text, number, time, and calculated.

- *Views*

Views are user-defined ways of looking at information in a database. They are roughly analogous to reports that might be created in a relational database, but with far more flexibility. Different views may be created based on the reporting and viewing requirements of the application. With Notes 4.5 there are two different types of views, the Standard Outline and the Calendar view. The Standard Outline is the traditional type of Notes view that displays information from documents in columns. The Calendar view displays time and date driven information in a diary format.

Developers can allow users to dynamically sort the columns in a view. Developers may also customize the graphical appearance of views.

- *Navigators*

Navigators are a graphical “table of contents” for a database. For example, the navigator for a regional sales database might be a map of that region. By using navigators, it is possible to provide your users with a business-specific user interface for a Notes application. For example, users can click on a set of maps to find documents concerning a specific region or territory.

Remote and mobile use of a document-oriented database means that users throughout the network may be creating and sharing the information in Notes documents. Notes ensures that all users have the same and latest version of a document through the process of replication. Replication ensures that all copies of a Notes database are synchronized over time. Replication is a fundamental feature of Notes that allows not only for the synchronization of data, but also of applications.

Notes databases store information, not just discrete data as stored by other types of databases. Relational database management systems (RDBMSs) store data in a highly structured format. Notes databases allow for the creation of compound documents that can include both structured and unstructured information.

Notes and RDBMSs Are Complementary

Notes is not a relational database. This is a key distinction since Notes does not provide capabilities usually associated with RDBMSs, like referential integrity and distributed transaction support. However, Notes and RDBMSs are not mutually exclusive; in fact, they can exchange data and thereby extend each other, creating the possibility for very powerful

applications. For example, a manager might use Notes to create a monthly business report. Part of that report would pull in figures from a monthly expense database using an interchange tool.

Here are two key differences between an RDBMS and Notes:

- Real-time access to data.

While the RDBMS keeps a unique version of the data at a given time, Notes may have different versions of the documents in databases distributed across the network. This is why Notes is best suited for applications where the immediate availability of current data is not a strict necessity.

- Locking a record or table.

A Relational Database Management System can lock a record or table while the user reads or updates the data. This is not supported by Notes. Since a Notes database can be distributed across workstations and networks, it is possible that different users may edit the same document at the same time. However, Notes can handle the replication conflicts. Properly designed Notes applications rarely suffer from the lack of a locking mechanism as this is a facility unique to transaction-oriented RDBMSs.

The LotusScript:Data Object (LS:DO) supplied with Notes Release 4.5 is one available data integration method. LS:DO objects can access any database for which an Open Data Base Connectivity (ODBC) driver exists. Accessing a database consists of three steps: making the connection, specifying a query, and getting values from the result set. LS:DO has three object classes that relate to these functions. The LS:DO can be used on both Notes clients and servers.

In addition, there is a wide variety of other data integration tools available from Lotus and its Business Partners, which provide server-to-server data interchange interfaces for relational databases such as Sybase, Oracle, and DB2/2. Lotus NotesPump™ is an example of server-based, scheduled data interchange. The MQSeries™ Link for Lotus Notes allows integration between transaction systems such as CICS and IMS/DC and the Notes object store.

The Messaging Infrastructure Enables a New Class of Application

Notes databases are animated by the messaging infrastructure. Information is not just stored in or retrieved from databases in a bi-directional exchange between user and application. Database information can be routed between users or even other databases. The Notes messaging infrastructure consists of a transport back-end that runs on almost any wiring topology or network operating system. The messaging transport enables enterprise-wide applications such as:

- *Workflow applications*

Workflow applications automate the task of managing information within workgroups. For example, a proposal-writing workflow application would let users write a proposal then route a request for the reviewer to review and make comments within the document. When the proposal is updated, another request would be sent out to ask for sign off on the final document.

- *Forms-routing applications*

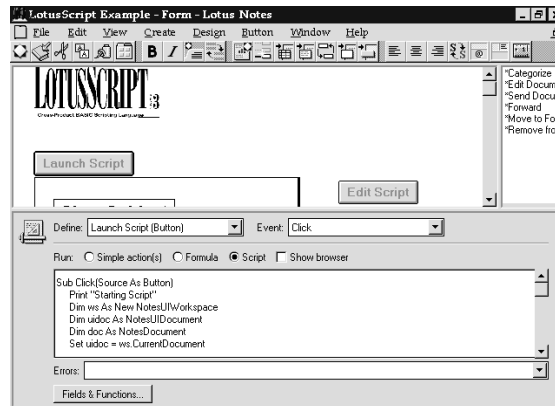
A forms routing application would focus on sending a form for a specific action. For example, a travel authorization application would send the travel form to the appropriate manager for electronic approval.

- *Messaging-reliant applications*

Some applications may be focused almost entirely on the routing of messages. A scheduling application would manage the routing of messages to set the time for a meeting.

Notes Has a Cross-Platform, Structured, BASIC-Compatible Programming Language

Notes comes with many types of design elements which are used to create a range of application types. The Notes Integrated Development Environment (IDE) is the single interface to all of the Notes application design elements. You need to learn how to use the IDE only once to access all Notes design features. The IDE displays a three-pane work window. The design pane in the top left half shows the form or view that is being created. The smaller action pane on the right lists available actions. The programmer pane below is where the developer creates the code. Here is an example of the IDE three-pane work window:



12 Lotus Notes Release 4.5: A Developer's Handbook

Notes design elements include:

- *Forms and Views*

Developers create forms and views as the basis of Notes application development. A technically sophisticated user can also create forms and views for a database with relative ease. These are the most simple, and fundamental, design elements.

- *Simple Actions*

Simple actions are useful precoded “units of work.” An example is “Send Newsletter Summary” which, when combined with the full-text search capability of the Notes database, allows messages containing document links to be sent to any number of users. Simple actions combine Notes functionality into elements that can be used in any Notes application.

- *LotusScript*

Developers can use LotusScript to program sophisticated applications. LotusScript is a superset of the familiar BASIC language. Key enhancements over standard BASIC include object-orientation and extension for event-driven environments.

LotusScript is also multi-platform. Notes applications that contain LotusScript run on all Notes Release 4.5 clients and servers, no matter which operating system the application was originally developed on.

The LotusScript programming environment has a script editor, debugger, and variable / property inspector.

Notes Release 4.5 provides access to the Notes Object Model, which is a set of classes and their associated methods, properties, and events that you can access from the Script editor window.

The Object Model can be browsed in the Script editor using the integrated browser. The browser provides a listing of the following categories of components: Notes classes, Notes constants, Notes subroutines and functions, Notes variables, and the LotusScript language. You can select a component from the browser list box, and open up component categories by clicking the arrows.

In Release 4.5 there are new events, in addition to the usual open and close events, that can be captured to support the drag and drop functionality of the calendar view.

RegionDoubleClick	QueryPaste
QueryOpenDocument	PostPaste
QueryRecalc	QueryDragDrop
QueryAddToFolder	PostDragDrop

- *Formulas and @Functions*

Formulas consist of variables, constants, @Functions, operators, and keywords. @Functions are used to perform specific tasks within Notes. For example, @Created displays the create date of a document. @Round (number) rounds the designated number to the nearest whole number. There are over 150 @Functions available to applications developers.

- *Design Templates*

Notes comes with a set of design templates for the most common business applications. Using these templates can save significant development time and money.

- *Navigator and HotSpots*

The Navigator allows advanced developers to create a graphical “table of contents” for a database. Navigators can contain HotSpots which initiate actions.

- *Actions*

The user interface contains an Action Bar which displays the predefined or customized action buttons that developers choose to incorporate into the application. System-provided actions include Print, Send, and Close. Customized actions are programmed by the developer and can trigger more sophisticated processing and logic. Customized actions may be created using LotusScript or the macro language.

- *Agents*

Agents, which are similar to macros, are created to automate time or event-driven processes. You identify the name, the event trigger, documents to act on, and the action itself to create a simple agent.

You Can Leverage a Range of Development Skills

The beauty of the Notes ADE is that a range of application design elements is available to everyone from sophisticated users to highly experienced programmers. Applications can be as simple or sophisticated as the need demands and resources provide. Companies without a lot of IS support can still develop useful applications.

Major Enhancements of Release 4.5

In the following chapters, we will go through the different steps that are needed to design and develop an application using Notes Release 4.5. We will not cover the Release 4.5 server aspects except when it has implications to the design of the database you develop.

To give you an early flavor of the features that have been added to Release 4.5, here is an overview of just some of the new features:

User Interface

- Background bitmaps
- 3D Text
- Colored tables
- Date and Time controls

Calendaring & Scheduling

- Allows native Notes calendaring and meeting scheduling with other users
- Free-Time database
- New calendar view style
- Two day, one week, two weeks, one month formats
- New calendar time controls

Enhanced IDE

- Context-Sensitive help from the browser
- Color and font support for LotusScript keywords
- Search and Replace in the IDE

LotusScript

- Support for LotusScript 3.1 new commands
- Script Libraries
- New Notes classes for NotesForm, NotesName, NotesDateRange, NotesInternational, NotesTimer, NotesUIDatabase, NotesUIView
- Updates to existing Notes classes, including:
NotesUIDocument.*EditDocument*, NotesAgent.*Run*,
NotesDatabase.*DelayUpdates*, NotesDocumentCollection.*FTSearch* and
NotesDocumentCollection.*StampAll*
- Database-level LotusScript modules
- Import/Export of LotusScript from the IDE

New @Functions

- Additional functions for the @function fans

Security Enhancements

- Workstation security with Execution Control Lists (ECLs)

Internet Enhancements

- Integrated Notes HTTP server
- Personal Web Client including Web Ahead and Page Minder agents
- Background bitmaps on forms
- Java support
- Net.Action — Internet site creation tool

Chapter 19

Accessing Transaction Systems Using MQSeries



This chapter describes the MQSeries Link for Lotus Notes product line. It covers the following main components:

- IBM MQSeries™ link for Lotus Notes and CICS™ link for Lotus Notes
- IBM MQSeries™ link Extra for Lotus Notes
- MQSeries link for Lotus Notes Extension (MQLSX)

These components enable the integration of Notes environments with enterprise transaction oriented systems like CICS and MVS.

About MQSeries

A solution that effectively integrates transaction systems with new client/server systems is required. It needs to preserve and leverage the strengths of each platform with minimal tradeoffs in functionality and ease of use:

- Support for transaction processing monitor system
- Single environment with transparency of platforms
- Reliability
- Mobile support
- Cost effectiveness
- Support for standards
- Accessibility

MQSeries products enable applications to use message queuing to participate in message-driven processing. With message-driven processing, applications can communicate across the same or different platforms, by using the appropriate message queuing software products. For example, MVS/ESA and OS/400 applications can communicate through MQSeries for MVS/ESA and MQSeries for OS/400 respectively. With MQSeries products, all applications use the same kinds of message headers; communications protocols are hidden from the applications. There is an ever-growing number of platforms on which MQSeries is supported

(currently about 20). Once connected through the respective links, the different operating systems, networks, etc. are transparent to the user of the application.

MQSeries products are designed for assured message delivery. Processing is such that when messages are being transmitted to remote queue managers, the messages are moved in discrete transaction units, or batches, where confirmation of receipt is always obtained before a particular message is deleted at the transmitting queue manager. To achieve this, the sending and receiving ends of the link commit batches of messages in unison.

Where Does Lotus Notes Fit?

Lotus Notes applications are designed to manage unstructured data. Many organizations have identified the requirement to integrate Notes with host-based transaction processing systems. This has several advantages:

- Extends access to existing host transaction applications to Notes
- Provides users with a single point of access to these systems and to client/server applications

There are three different forms of integration technology currently available to achieve this purpose:

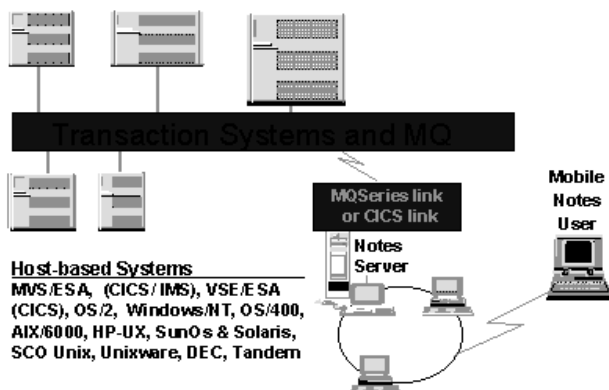
1. User-initiated transaction access:
IBM MQSeries™ link for Lotus Notes, CICS™ link and CICS link extra for Lotus Notes
2. Host-initiated transaction data download:
IBM MQSeries™ link Extra for Lotus Notes
3. LotusScript Extension:
MQSeries link for Lotus Notes Extension (MQLSX).

These technologies remove the constraints of partial solutions, providing an IS organization with robust, transaction-oriented solutions for integrating their applications systems with Notes. These components leverage existing transaction processing systems, requiring no changes in technology or business processes to conduct backup and recovery, logging and auditing, system measurement, workload balancing or performance monitoring. System security also remains unchanged.

MQSeries allows Notes to participate in transaction systems by allowing Notes to initiate transactions and by acting as a store-and-forward data repository for data from transaction systems. The enterprise critical data and the associated business rules continue to be managed by the transaction system. This allows Notes applications to take advantage of both the data storage and processing logic of very large, distributed transaction systems.

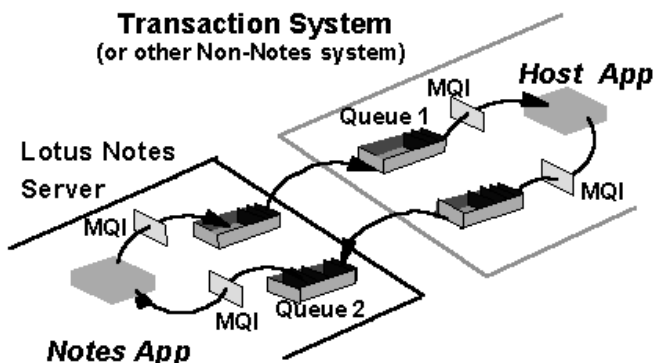
The following figure shows the integration of Lotus Notes to the host environment. The Notes user can be on a LAN-attached workstation or on a disconnected laptop which stores transactions and periodically attaches to the server to “send” the transactions through MQSeries Link to the host.

An Extended Transaction Model Approach



Transactional Overview

When a Notes application accesses MQSeries Link, the following function flow occurs across the platforms which is transparent to the user.



Applications for MQLink

Any data or proprietary processing logic accessible by a transaction program can be accessed by Notes using MQSeries. The MQSeries API, called the Messaging Queue Interface(MQI), gives Notes access to any logic or storage available on the target system, including non-relational DBMS file stores like ISAM and sequential files.

The MQSeries link also enables Lotus Notes clients and servers to work with multiple systems simultaneously, such as accessing a transaction running on HP-UX Tuxedo and a program running on DEC VAX concurrently. MQSeries link provides store-and-forward message queues for Notes application developers to utilize in Notes applications, which eliminates the necessity for developers to code network-specific application calls.

Technical Advantages

The MQSeries link for Lotus Notes functions as a special link from Notes to transaction systems. It has the following advantages over other middleware products:

- Integration with the Notes application development environment.

All the definition, design and testing take place in the Notes development environment. One of the implementations of MQSeries link for Lotus Notes is as a LotusScript Extension (LSX). LotusScript extensions expose their functionality and classes to LotusScript in exactly the same way as Notes itself does. Notes developers therefore have seamless access to Notes and MQSeries functionality.

- Application location transparency for developers.

The MQSeries link shields the Notes application developer from the multi-vendor, multi-protocol complexity of today's business networks and provides application-location transparency. The MQSeries link provides a programming interface for computers and networks from multiple vendors and offers a simple, reliable means of building distributed and client/server applications.

- Integration into Notes application interface.

Notes applications can transparently integrate transaction system data. LotusScript allows the data returned from host transactions to be posted directly to the Notes user interface.

- Time-independent (asynchronous) processing.

MQSeries allows time independent (asynchronous) processing, which means that when a message is created to initiate a transaction, that message does not have to be delivered immediately (if, for example, the

system the transaction runs on is not available at the time the message is created). MQSeries will assure that the message is kept until the transaction can process it. Using MQSeries and the agent capabilities of the Notes server, it is possible to develop incredibly sophisticated applications, such as future point-in-time workflow applications for scheduling production runs for customer orders received in Notes.

- Communication through queues.

All communication using MQSeries occurs through queues only. The MQSeries link couples queued, store-and-forward messaging with Notes' powerful integrated client/server messaging to deliver a unique set of application development capabilities unmatched any other product set.

- Data Integrity Protection.

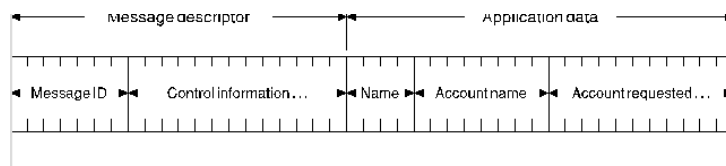
MQSeries allows access to enterprise *processes*, whereas the other techniques allow access to enterprise data. Using MQSeries, the *only* way to access data is through a transaction, never directly. This may be considered limiting in that programs must be used to access data. Yet, it is enabling in that using programs to access data maintains data integrity, whereas accessing the raw data directly from external applications could put that data integrity at risk.

Terminology

This section covers some important terms to understand when dealing with transaction systems.

Message

A message is a string of bytes that has meaning to the applications that use the message. In MQSeries, messages have two parts, a message descriptor and application data. The content and structure of the application data are defined by the application programs that use them. The message descriptor identifies the message and contains other control information or attributes, such as the date and time the message was created, the type of message, and the priority assigned to the message by the sending application. The message descriptor and application data are shown as separate parts. Information that is specific to the application, such as <Account name> in this example, is in the application data part of the message.



Queue

In physical terms, a queue is a type of list that is used to store messages until they are retrieved by an application. Local queues exist independently of the applications that use them. Each queue belongs to a queue manager, which is responsible for maintaining it. The queue manager puts the messages it receives onto the appropriate queue.

In MQSeries, messages can be retrieved from a queue by suitably authorized applications according to these retrieval algorithms:

- First-in-first-out (FIFO)
- Message priority, as defined in the message descriptor. Messages having the same priority are retrieved on a FIFO basis
- A program request for a specific message.

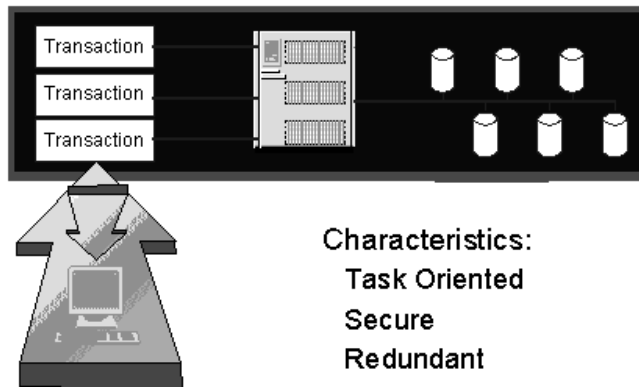
Queue managers

A queue manager is that part of an MQSeries product that provides the messaging and queuing services to application programs, through the Message Queue Interface (MQI) program calls.

Transaction Oriented Systems

Transaction oriented systems are reliant on services from a transaction processing monitor. These services provide rollback, backup, recovery, logging and auditing.

The following figure shows a typical transaction oriented system environment where a terminal is used to enter transactions to a host-based CICS application for example:



MQSeries Link and Link Extra for Lotus Notes and CICS Link and Link Extra for Lotus Notes

Overview

MQSeries link and link extra, and CICS link and link extra are tasks that enable the exchange of data between Notes and a set of APIs (MQI in the case of MQSeries and ECI for CICS). This link technology also controls the flow of data between the Notes application and the transaction system. The actual translation, connecting, delivery and reply from the target system are all under the control of the link technology. Notice that a single user request from a Notes client or a Web Browser can generate multiple requests to one or many target systems, allowing work to be processed in parallel.

CICS link and CICS link extra for Lotus Notes connect directly to CICS, which runs natively on AIX, OS/2, OS/400, MVS/ESA, VSE/ESA, Windows NT, HP, and DEC. A host-based system with CICS requires no additional host software to integrate with Notes. To access a CICS application using the MQSeries link or MQLSX, you need MQSeries on the target system in addition to CICS. MQSeries is not required for use with the CICS link or the CICS link extra.

MQSeries provides messaging and queuing support, routing the message to the appropriate target system in the network so that it can either be accessed by programs (for example: a Lotus Notes server) servicing these queues; or alternatively, the arrival of the message can trigger an application process or transaction. It supports application to application connectivity through queues in an asynchronous, loosely coupled model, as opposed to dependency on customized, tightly-coupled connections between platforms and application programs.

A Notes application consists of a variety of objects, one of which is the Notes form, which in turn contains several types of fields. Some of these fields are populated with data obtained from a transaction processing system or other non-Notes systems. MQSeries link and link extra for Lotus Notes (or CICS link or link extra for Lotus Notes) provide an extension to a field definition describing how the data can be obtained or updated.

MQSeries and CICS link run as server add-in tasks, while the MQSeries and CICS link extra run as independent processes.

MQSeries and CICS Link for Lotus Notes

The MQSeries or CICS link provides the following services to an application developer and the system in operation:

- It manages the fields it receives from the Notes application.
- It handles all security for access to the transaction systems as well as security as the result of those transactions
- It handles all error conditions that result from the interaction with the transaction system.
- It switches the Notes data to a transaction system or switches the transaction back to Notes.
- It handles all interaction with the transaction system.
- It notifies the Notes application as to completion of its task
- It controls the update to any Notes database.

Application Development

The development of the actual application is not much different from any other Notes development effort. All the definition, design and testing take place in the Notes development environment.

To access transaction systems or non-Notes system data, the Notes developer needs to know the key fields that are used in the host system to access the data and what fields they wish to have accessed. The linkage technology provides a programming interface for computers and networks from multiple vendors and offers a simple, reliable means of building distributed and client/server applications. The link shields the Notes application builder from the multi-vendor, multi-protocol complexity of today's business networks and provides application-location transparency.

For example, a Notes application is being developed to handle customer service. The basic demographic information (customer name, address, phone number, and so forth.) for that customer is stored in a transaction system and is keyed by their customer number. The Notes application developer would define fields for the customer name, address, and so forth. During the definition process the developer mail-enables those fields or the form itself to either CICS link or MQSeries link. This means that the fields will be sent to the link when the form is saved or updated.

Where the application development process does differ from traditional Notes application development is in transaction mapping. In the transaction mapping definition, the developer creates a control record in the MQSeries link Application Transaction Map (MATM) database. This definition stage

is somewhat similar to the establishment of a connection record within Notes. To define the transaction mapping, the developer needs to define some processing definitions:

This “connection” record defines:

- The actual transaction system, for example CICS™, IMS or non-Notes system (for example, OS/400®, DEC™ VAX™ VMS™).
- What Notes fields map to what fields on the defined system.
- Proper security.
- What actions are permitted on the data (adds, deletes, updates).
- How the results are processed.
- Any other fields needed for processing.

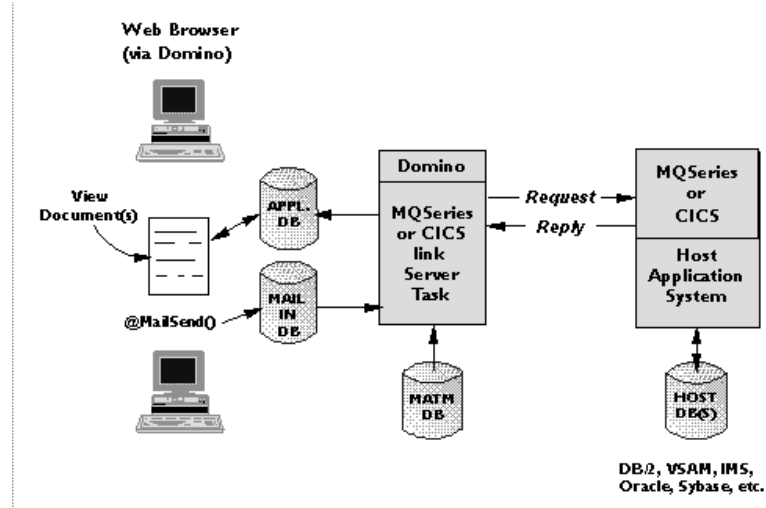
The actual link and switching process is managed by controlling the MATM database. This is a Notes database using standard Notes conventions. It controls the link that is a Notes server add-in task, so its management is most appropriately performed by the Notes Administrator.

Transaction Flow

A typical transaction in an application integrated with MQSeries and CICS link works like this:

1. The Lotus Notes @MailSend function is used to send key information along with fields to a Notes mail-in database managed by MQSeries link.
2. MQSeries link extracts the key along with the requested fields and maps these to an MQSeries request.
3. MQSeries link then plugs in the security, transaction information (keys, fields, and so forth.)
4. The request is then issued to MQSeries.
5. MQSeries then sends the request to the appropriate system using its routing technology.
6. When the transaction is complete, the requested data is placed in an MQSeries message and sent back to MQSeries link.
7. MQSeries link searches for returned responses in addition to processing outgoing tasks. The incoming data and return codes are processed by MQSeries link and then either posted to a holding database or switched back into the original application database
8. The user is then notified that the information has been returned.

The following diagram shows the different components and the flow of a transaction. The possibility of having an application which is served over the Web through Notes Domino technology is also depicted.



MQSeries Link Setup

Setup of the MQSeries link is done through the following procedure:

1. **MQSeries**
Install the MQSeries product for the operating system being used. The appropriate system queues also need to be designed and setup.
2. Setup the connectivity between the MQSeries on the Notes Server and the transaction system.
3. Add the MQLINK add-in task to the Notes server initialization file and restart the server.
4. Create the Mail-in database and add to the server address book.
5. Copy the MQSeries Link Notes databases (.nsf) from the MQSeries directory to the Notes Data directory.
 - MQLINK.NSF — used to map data to and from Lotus Notes
 - AMQSAMPL.NSF — sample application using @MailSend

6. Setup the mapping between documents and messages in the Link database using the External Call Parameters Form:
 - Enter the Name of the External Call Parameters entry and the Mail-in database name and form:

External-Call Parameters

Entry: ⑆ MQENTRY ⑆

Database Information

User Database Name : ⑆ AMQSAMPL ⑆

Form to update : ⑆ Sample ⑆

7. Enter the Request and Reply offsets for the mapping of the data.
Request offsets is the layout of the @MailSend() message arriving on the Mail-In database. The field names are not used, but the order of the field definitions is significant. Reply Offsets define the layout of data returned in the MQSeries message to the reply queue read by MQLINK.

The screenshot shows a Lotus Notes window titled "(Untitled) - Lotus Notes" with a menu bar (File, Edit, View, Create, Actions, Text, Window, Help). The main text area contains the following content:

Request Offsets
 Syntax:
 'FieldName <space>Start <space>End <space> CHAR |
 S390-BINARY | INTEL-BINARY'
 ⑆ function 0 2 CHAR
 Msg 3 104 CHAR ⑆

Reply Offsets
 Syntax:
 'FieldName <space>Start<space>End <space> CHAR |
 S390-BINARY | INTEL-BINARY'
 ⑆ RData 0 101 CHAR
 Time 102 127 CHAR ⑆

At the bottom of the window, there is a label "Form to update" next to a blank line.

8. Enter the Message Queuing Parameters and error handling entries. The Message Queuing Parameters are:
 - Request Queue Name is the Outgoing message queue; that on which MQLINK will place the message for the MQSeries application.
 - Reply Queue Name is the incoming message queue for data from the MQSeries application to MQLINK. MQLINK will read this to update the Lotus Notes document.

- Message Format Field is the data type of the MQSeries message.

(Untitled) - Lotus Notes

File Edit View Create Actions Text Window Help

Message Queueing Parameters

Request Queue Name : SYSTEM.SAMPLE.NOTES.INQUEUE
 Reply Queue Name : SYSTEM.SAMPLE.NOTES.OUTQUEUE
 Message Format Field : MQSTR

Error Handling

Syntax:
 'Error Condition Value <space> start-offset <space> end-offset
 <space> CHAR'

In the event of an error you may provide additional information:
 Syntax:
 'Fieldname <space> start-offset <space> end-offset<space> CHAR'

Field Name, buffer start and buffer end

9. Set up your application to create mail entries to your Mail_In database as specified below. A button can be placed on a form of your application to send data to MQSeries as in the sample application:

(Untitled) - Lotus Notes

File Edit View Create Actions Text Window Help

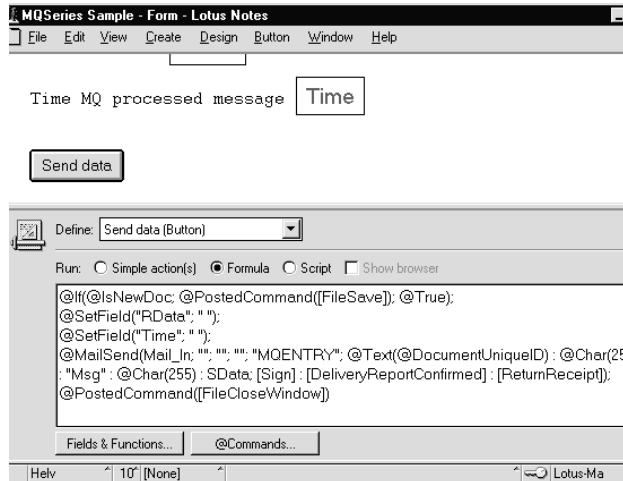
The data typed in the *data to be sent* field will be sent to the **Mail_In** database when the *Send data* button is pressed. It will then be read from the **Mail_In** database by MQLINK and placed on the MQSeries target queue specified in the **Link Database** document MQENTRY. The message will then be retrieved by AMQSLNK0, the text tokenised, reversed and placed on the *reply* queue. MQLINK will then remove this reply message and update this document.

Enter target Mail-In database name Mail-In Database
 Enter data to be sent Hello world !
 Data received
 Time MQ processed message

Send data

Received data

The code in the Send button is used to forward the message to MQSeries through the Mail_In database for processing:



MQSeries and CICS Link Extra for Lotus Notes

As described above, MQSeries and CICS link allow Lotus Notes users to initiate the interaction with an MQSeries-connected or CICS application.

However, since transaction systems are typically event-driven and an event will be triggered by some outside occurrence, it may be required that the host initiate notification of an external event or some changed data to one or more Notes applications. Take, as an example, a client submitting a change of address, or a supplier issuing a price change.

With the new functionality provided in MQSeries and CICS link Extra, the host transaction system is able to initiate an unsolicited transfer of transaction data to the Notes server. This data may then be used to update or add one or more documents to a Lotus Notes database (document updates may also span multiple databases).

MQSeries or CICS Link Extra provide the following services:

- All interaction with the transaction system.
- Handling of any error conditions that result from the interaction with the transaction system.
- Control of update(s) to the Notes database(s).

- Searching for matching documents based on the key or keys defined in the MATM database.
- Managing and mapping the fields it receives from the host application.

Application Development

The development of the actual application differs from that described for MQSeries link and CICS link, above, in that more of the design and development effort are related to the host application system. The Notes development effort is limited to the definition and system testing of the application solution.

In order to update Notes application documents with transaction or non-Notes system data, the Notes developer needs to know the key fields that are used in the host system, and what documents / fields will be updated under what rules.

Using the customer service example from MQSeries / CICS link, above, the host application system might initiate an update to the Notes application. The basic demographic information (name, address, phone number, and so forth), for that customer is stored in a transaction system and is keyed by that customer number. The Notes application developer would identify the key field as Customer Number, and the rule for updating the document. For example, a rule might be — “If Found, Update.” This would cause the link Extra Task to update the customer document with the data received from the host application when the Customer Number received is the same as that in the document.

Managing the Link Extra Process

The MQSeries link extra for Lotus Notes Link-database definition is an extension of the MQSeries link for Lotus Notes Link-database. As a result the MQSeries link extra for Lotus Notes Link-database may be used to define entries for both MQSeries link and MQSeries link extra.

In addition to those fields used by MQSeries or CICS link, the link extra component includes definitions for:

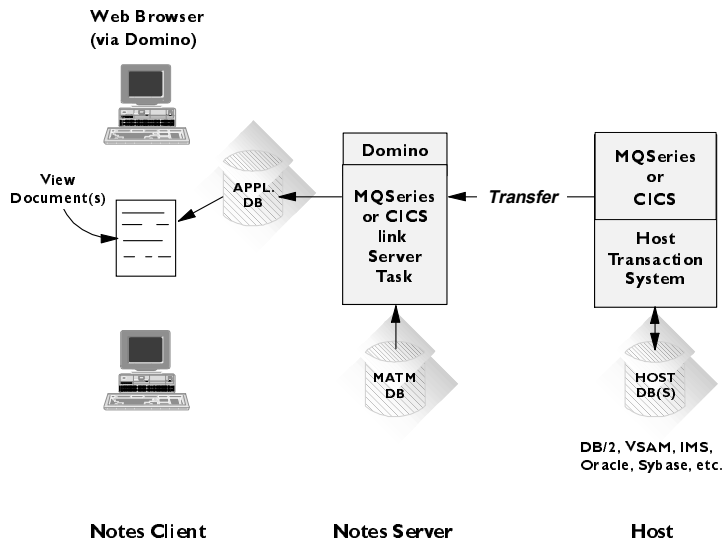
- The primary rule used to define how the data is processed given existing documents in the Lotus Notes database
- An optional Additional Selection Formula used to specify search criteria to be used against the Notes database

An exit is also provided for special error handling, which the user can tailor to meet the specific needs of their applications, if required.

A Typical Host-Initiated Transaction

As in MQSeries link or CICS link, the Notes Administrator has set up the proper security, transaction identification and mapping to the target system during the implementation phase. In addition, he or she has identified the key field(s) and processing rule(s) that will control the update process. The host application system generates message data for the Notes server containing information to be updated within one or more Notes documents. This data will include key elements identifying documents to be updated. The data is placed in an MQSeries message or CICS commarea and then sent to the appropriate system using MQSeries or CICS. The arriving data then triggers the link extra task, MQLINKX. The incoming data and return codes are processed by the task, which extracts the key, as defined in the MATM, and searches the Notes application database for any and all matching documents. The task will then take the appropriate action, for example, update or insert a document, based on the rules defined in the MATM for that application.

The heart of the Link Extra product is the MATM notes database just as in the MQSeries and CICS link, above. It provides mapping between the transaction system application and fields defined in the Lotus Notes application databases.



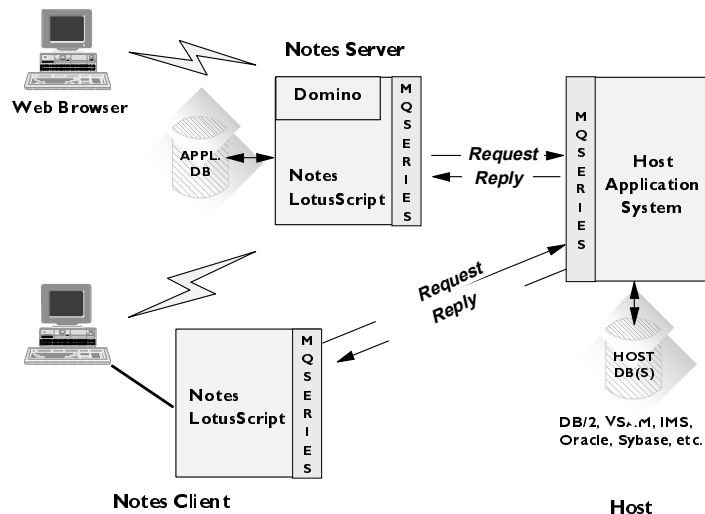
The MQSeries Link LotusScript Extension (MQLSX)

The MQSeries link LotusScript Extension enables a Notes application to interact with non-Notes applications throughout the enterprise via MQSeries. It provides direct integration between Lotus Notes and MQSeries software, extending the scope of Notes to include data and transactions that are part of other environments, and it gives the Notes LotusScript application the ability to initiate an application transaction to process or request information from any of the enterprise systems that can be accessed through MQSeries.

It differs significantly from the Link and Link Extra components described above. Whereas they are implemented by providing definitions in turn used by agents or independent tasks, the MQLSX is an Application Programming Interface that is called from LotusScript to interact directly with MQSeries through its MQI (Message Queuing Interface).

Another significant difference is that the MQLSX can be implemented on either the Notes server or the Notes client, enabling the LotusScript application to access the Message Queue Interface (MQI), and subsequently the MQSeries infrastructure, directly, when such a design is required by the business application: for instance to allow control at the Notes client for an application that requires steps to be completed sequentially rather than in parallel.

It requires an MQSeries environment and a corresponding MQSeries application with which to communicate.



The MQLSX does not make any calls to Notes. The applications handle what is updated in Notes, splitting the messages received from MQSeries into fields, and adding them to new or existing Notes documents.

The MQLSX can only be used with Notes Release 4. To run the MQLSX in a Notes Server environment you need at least one of the following installed on your server:

- MQSeries for OS/2 Version 2.0.1 or later
- MQSeries for Windows NT Version 2.0 or later

Note MQLSX implementations on Windows 95 and various UNIX platforms will soon be available.

To run the MQLSX in a Notes Client environment, you need at least one of the following installed on your workstation:

- MQSeries Client on OS/2
- MQSeries Client on Windows 3.1
- MQSeries Client on Windows NT

Note The MQSeries Client requires access to at least one MQSeries Server.

Setting Up Your MQLSX Environment

Before you run a script using the MQLSX, check that you can run the queue manager that your script will connect to, and that the necessary queues are in place. You can do this by using the command DISPLAY QMGR; this command fails if the queue manager is not running.

On Windows NT and OS/2 the MQLSX dynamically detects and uses either the MQSeries server DLL (mqm.dll) or the MQSeries client DLL (mqic.dll). The MQLSX searches for and uses these in the order mqm.dll, then mqic.dll. The search uses standard system services to check for the DLL in your current working directory before searching your normal DLL search path.

Note This is in the process of being changed.

Tip If you are in doubt about which .DLL the MQLSX is using, you can run the MQLSX with Trace on and look for the entry under EstablishEPS.

MQLSX Classes

The LotusScript / MQI interface is supplied as a LotusScript Extension Module (LSX) that provides the following classes:

- MQGetMessageOptions

This class encapsulates the various options that control the action of getting a message from an MQSeries Queue.

- **MQMessage**
This class represents an MQSeries message. It includes properties to encapsulate the MQSeries message descriptor (MQMD), and provides a buffer to hold the application-defined message data.
The class includes methods (Write methods) to copy data from a LotusScript application to an MQMessage object and similarly methods (Read methods) to copy data from an MQMessage object to a LotusScript application. The class manages the allocation and deallocation of memory for the buffer automatically. The application does not have to declare the size of the buffer when an MQMessage object is created as the buffer grows to accommodate data written to it.
- **MQProcess**
This represents an MQSeries process definition object (used with triggering). Using the MQLSX, you can interrogate the properties of the process definition object within your script.
- **MPutMessageOptions**
This class encapsulates the various options that control the action of putting a message onto an MQSeries Queue.
- **MQueue**
This represents a connection to an MQSeries Queue. This connection is provided by an associated MQueueManager object. When an object of this class is destroyed, it is automatically closed.
- **MQueueManager**
This represents a connection to a queue manager. The queue manager may be running locally (an MQSeries server) or remotely with access provided by the MQSeries client. An application must create an object of this class and connect it to a queue manager. When an object of this class is destroyed, it is automatically disconnected from its queue manager.
- **MQSession**
This is the root class for the MQSeries link LotusScript Extension. There is always one and only one MQSession object per LotusScript instance. An attempt to create a second object will create a second reference to the original object.

Using the MQLSX

When designing a LotusScript application that uses the MQLSX, the most important item of information is the actual *message* that will be sent or received from the remote MQSeries system. Therefore you must know the format of the items that will be inserted into the message.

You should also know:

- The codepage that the remote system runs in
- The encoding that the remote system requires

To help you to keep your code portable it is good practice always to set the codepage and encoding even if these are currently the same in both the sending and receiving systems.

Notes Server or Notes Client

When considering how to structure the implementation of the system you design, remember that your MQLSX scripts *must* run on the same machine that you have MQSeries installed on. If you do not have MQSeries installed locally, make use of the remote agent function available within Notes. You will need to do this explicitly, as Notes by default will run a script locally.

When your MQLSX script runs, it will need an MQSeries application that picks up the message your script has sent or one that puts a message on a queue for your script to get. For this to work, both your script and the MQSeries application need to know the structure of the message they are dealing with.

Using Large Messages

The MQLSX supports messages up to 4MB long if memory is available. However, when using large messages Notes restricts:

- Plain text to 64K.
- Fields added to the summary buffer to 32K.
- LotusScript string length to 32000 characters.

Tip To overcome this limitation consider appending strings into a rich text field in Notes, which has no size restrictions. The MQLSX has been designed to take advantage of this feature. Copy the data 32,000 characters at a time from an MQLSX message into LotusScript strings. From the LotusScript strings, append the data into a rich text field. For example, where the message data is greater than 32K (the maximum length of a string in LotusScript), read the data in multiple parts.

This code fragment assumes that the message, MyMsg, has already been taken from the queue using the GET method of the MQQueue class and is less than 64K in length:

```
Dim MessagePartA As String
Dim MessagepartB As String
...
...
MessagePartA = MyMsg.ReadString(32000)
MessagePartB = MyMsg.ReadString(MyMsg.DataLength)
```

Embedded Nulls in a String

The MQSeries constants, used for the initialization of three MQMessage properties below are not supported by the MQLSX. The LotusScript String call allows you to do the same thing.

MQMI_NONE (24 NULL characters)

MQCI_NONE (24 NULL characters)

MQACT_NONE (32 NULL characters)

To set the MessageId of an MQMessage to MQMI_NONE:

```
mymessage.MessageId = String(24,0)
```

To set the CorrelationId of an MQMessage to MQCI_NONE:

```
mymessage.CorrelationId = String(24,0)
```

To set the AccountingToken property of an MQMessage to MQACT_NONE:

```
mymessage.AccountingToken = String(32,0)
```

Message Descriptor properties

Where an MQSeries application is the originator of a message and MQSeries generates the AccountingToken, CorrelationId, MessageId — you are recommended to use the AccountingTokenHex, CorrelationIdHex, and MessageIdHex properties if you want to look at their values, or manipulate them in any way — including passing them back in a message to MQSeries. The reason for this is that MQSeries-generated values are strings of bytes that have any value from zero through to 255 inclusive. They are not strings of printable characters.

Where your MQLSX script is the originator of a message and you generate the AccountingToken, CorrelationId, MessageId, you are recommended to use the AccountingToken, CorrelationId, and MessageId properties.

Receiving a Message from MQSeries

There are several ways of receiving a message from MQSeries:

- Polling by issuing a GET followed by a wait, using the LotusScript TIMER function
- Issuing a GET with the Wait option; you specify the wait duration by setting the WaitInterval property. This is recommended when, even though you set your system up to run in multi-threaded environment, the software running at the time may only run single threaded. This avoids your system locking up indefinitely.

Caution Issuing a GET with the Wait option and setting the WaitInterval to MQWI_UNLIMITED will cause your system to lock up until the GET call completes if the process is single threaded.

- Issuing a GET without the Wait option. In this case, once your script has issued the call, control is passed to the next script waiting to run. This second script, and any other scripts that may run before the original script regains control, must not affect any of the objects that the original script will expect to be the same as at the time it lost control.

When Your MQLSX Script Fails

Independently of the trace facility, for unexpected and internal errors, a First Failure Symptom report is produced.

You can find this report in a file named GMQnnnnn.fdc, where nnnnn is the number of the Notes process that is running at the time. You can find this file in the working directory from which you started Notes or the name of the path specified in the GMQ_PATH environment variable.

Example: MQSeries Link for Lotus Notes Extension

The MQSeries link for Lotus Notes supports initiation of a transaction from a LotusScript program. MQSeries provides a platform-independent application programming interface called the Messaging Queue Interface (MQI) available *directly* in LotusScript, allowing developers to create very advanced Notes applications that leverage transaction systems.

Using LotusScript, developers initiate a transaction either directly from the user interface (UI) or from a Notes agent on the Notes client and/or server. Through the MQSeries LSX, LotusScript can initiate a transaction plus allow the data returned from the host transaction to be posted directly to the Notes UI. If the end-user is viewing an in-memory version of the document, they must refresh their view to see the data returned from the transaction.

With the MQSeries LSX and the power of LotusScript, the Notes programmer can choose to return the transaction data to the UI, to the document, or to both. The script below establishes a connection to a transaction system that performs credit rating checks from within a Notes order-processing application. When a “Check Name Button” is clicked on a Notes form, the script will connect to MQ, open a queue, and put the Customer name on the queue. Then, the script will get the return message from the reply queue to determine the Customer’s credit rating.

```

*** Object Variables
Public uidoc As NotesUIDocument
Public workspace As NotesUIWorkspace
Public doc As NotesDocument
Sub Click(Source As Button)
*** Load the MQSeries LotusScript extension
    UseLSX "MQLSX"
    Dim Mqgms As New MQSession
    Dim Mqmqmgr As MQQueueManager

```

```

Dim MqqCheck As MQQueue
Dim MqqReply As MQQueue
Dim Mqpmo As New MQPutMessageOptions
Dim Mqgmo As New MQGetMessageOptions
Dim MQMsgSend As New MQMessage
Dim MQMsgRec As New MQMessage
Dim Msgdata As NotesItem

' ** String to contain the MQMessage data
Dim OutString As String

' **Connect to the MQSeries default QueueManager
Set MQQmgr = MQQms.AccessQueueManager("")

' **Setup CREDITCHECK as the output queue
Set MqqCheck = _
MQQmgr.AccessQueue("CREDITCHECK",0,"","","")

' **Setup REPLY as the input queue
Set MqqReply = _
MQQmgr.AccessQueue("REPLY",0,"","","")

' **Extract the name from the form and write it as
' **a string into the message object.
Set Msgdata = doc.GetFirstItem("CustomerName") _
MQMsgSend.writestring (Msgdata.Text)

' **Define the name of the queue to extract
' **messages from,
MQMsgSend.ReplyToQueueName = "REPLY"

' **Put the message on the output queue
MqqCheck.Put MQMsgSend, MQpmo

' **Get first message from the Reply queue
MqqReply.Get MQMsgRec, Mqgmo

' **Read the message in full, as a string, and
' **update form
OutString = _
MQMsgRec.ReadString(MQMsgRec.MessageLength)
Call doc.ReplaceItemValue ("CreditStatus", _
OutString)

```

End Sub