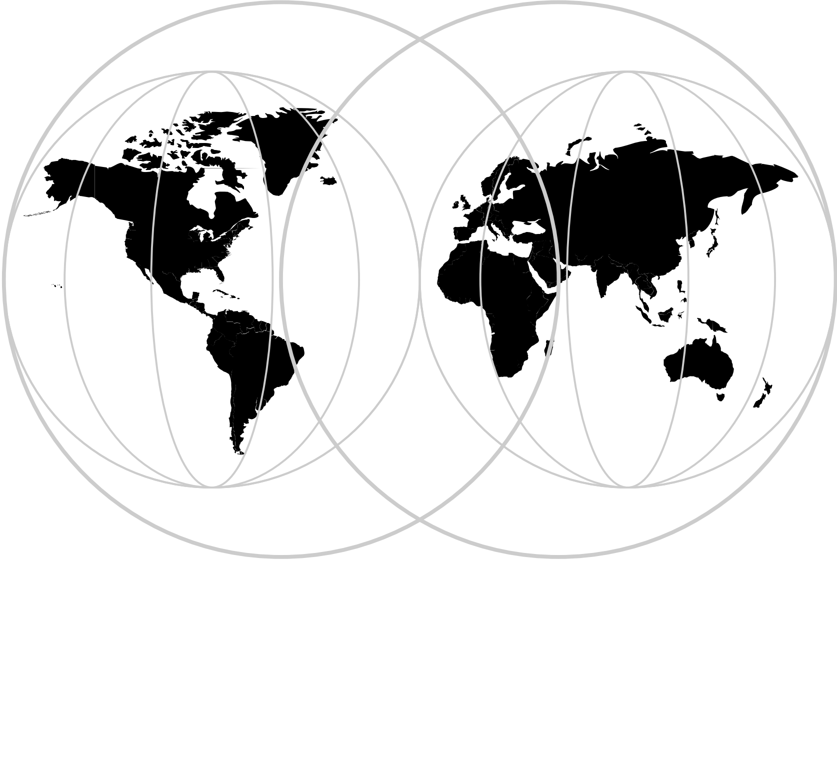
Lotus®

IBM

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.

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.

Contents

Chapter 5 Viewing the Database 87

Chapter 6 Programming in Lotus Notes 117

Chapter 16 Accessing Relational Database Management Systems With Notes 393

Chapter 17 Accessing Notes From Relational Database Management Systems and Query Tools 447

Chapter 18 High Volume Data Transfer With NotesPump 2.0 465

Chapter 19 Accessing Transaction Systems Using MQSeries 509

Chapter 20 Accessing Notes With the Notes C++ API 531

Appendix A Special Notices 565

Appendix B Related Publications 567

How To Get ITSO Redbooks 569

Index Index-1

Preface

This document describes how to develop applications with Lotus Notes Release 4.5.

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.

How This Document Is Organized

The document is organized as follows:

Part 1: Notes Application Development

- Chapter 1, “Getting Started”

  This chapter provides an overview of Lotus Notes and the Notes application development environment. It also lists the major enhancements of Notes Release 4.5.

- Chapter 2, “Creating Notes Databases”

  This chapter describes how to create new databases, or to use existing databases and modify them as needed.

- Chapter 3, “The Notes Integrated Development Environment”

  This chapter takes a close look at the integrated application development environment which allows you to create a variety of elements used in a Notes application.

- Chapter 4, “Designing Application Forms”

  This chapter illustrates how to design application forms based on the Document Library template provided with Lotus Notes.

- Chapter 5, “Viewing the Database”

  This chapter covers the structure of views, folders, and navigators.

- Chapter 6, “Programming in Lotus Notes”

  This chapter provides detailed information on the programming features available in Notes, in particular LotusScript.

- Chapter 7, “Using the LotusScript Extensions Toolkit”

  This chapter gives an overview of LotusScript extensions and introduces the toolkit that allows you to develop your own extensions.

- Chapter 8, “Using Agents”

  This chapter describes how to create an agent for task automation.

- Chapter 9, “Calendaring & Scheduling”

  This chapter discusses how to use the new LotusScript classes and @Functions related to Calendaring & Scheduling. It also covers how to create a view using the new calendar view type.

- Chapter 10, “Notes Workflow: An Example”

  This chapter gives an example of the Notes workflow capabilities by taking a close look at the Approval Cycle template provided with Notes.

- Chapter 11, “Working with Lotus Components”

  This chapter shows you how to work with Lotus Components. Topics such as creating components using LotusScript, Notes/FX, and NotesFlow publishing are covered.

- Chapter 12, “Notes Applications and Security”

  This chapter introduces Notes security from an application developer’s viewpoint.

Part 2: Extending the Reach

- Chapter 13, “Domino: Architecture and Configuration”

  This chapter discusses the Domino architecture and its elements.

- Chapter 14, “Domino: Creating Web Applications”

  This chapter introduces the Lotus Internet applications and describes in detail the Web application design elements.

- Chapter 15, “Domino: Sample Applications”

  This chapter takes a close look at three examples for using Lotus Notes with the Internet.

- Chapter 16, “Accessing Relational Database Management Systems with Notes”

  This chapter illustrates tools and techniques that can be used to access relational data resources from a Notes application.

- Chapter 17, “Accessing Notes from Relational Database Management Systems and Query Tools”

  This chapter covers NotesSQL.

- Chapter 18, “High Volume Data Transfer with NotesPump 2.0”

  This chapter provides information on NotesPump which is used to transfer data between Notes databases and external databases on different types of Database Management Systems.

- Chapter 19, “Accessing Transaction Systems Using MQSeries”

  This chapter describes the MQSeries Link for Lotus Notes product line.

- Chapter 20, “Accessing Notes with the Notes C++ API”

  This chapter documents how to use the Notes C++ API to access Notes facilities.

The Team That Wrote This Redbook

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

Marion Hawker is a Lotus Notes Specialist at IBM’s ITSO at Lotus Development, Cambridge, Massachusetts. She manages projects whose objective it is to produce redbooks on all areas of Lotus Notes, in particular the Notes application development features. Before joining the ITSO in 1995, she provided technical support for Lotus Notes and other products on a European basis. (Marion_Hawker@vnet.ibm.com; Notes e-mail: Marion Hawker/CAM/Lotus@Lotus)

Joe Arteaga is a consultant in the IBM Lotus Notes Practice and is based out of Toronto, Canada. His main role is the development and deployment of strategic solutions which are based on Lotus Notes. Joe has 10 years of industry experience, working with IBM and as a freelance consultant in the areas of workgroup computing and Client/Server. (jarteaga@vnet.ibm.com)

Gerald Krause is based at the IBM European Networking Center in Heidelberg, Germany, department of telecooperation. Currently, he focuses on the design and development of advanced technologies for workflow systems. Previously, Gerald worked on industry and research projects in the areas of distributed applications and their management. (krauseg@vnet.ibm.com)

Dave Morrison works for IBM in the United Kingdom. As a Lotus Notes application development consultant working for the world-wide Lotus Notes Service Line, he and his team are responsible for providing solutions to IBM customers in all aspects of Lotus Notes consultancy. (gbibmdam@ibmmail.com)

Hiroki Nakamura works for IBM Japan, Systems Engineering. He provides application development consultancy as well as training for systems engineers and customers of IBM Japan. (hnakamura@vnet.ibm.com)

Reinhold Strobl works for IBM Austria. As a member of the IBM Personal Systems Support and Services Center he provides support to customers in all aspects of application development. Before joining the IBM support organization he was a software designer and developer working out of an IBM software development laboratory. (rstrobl@vnet.ibm.com)

A number of people have provided support and guidance throughout the production of this book. In particular, we would like to thank **Alex Neihaus** from Notes Product Marketing, Lotus Development at Cambridge. In addition, we would like to thank the following people (in alphabetical order):

- Ned Batchelder, Lotus Cambridge
- Donna Carvalho, Lotus Cambridge
- Angela Finney, Lotus Cambridge
- Graphic Services, Lotus Cambridge
- Justine Grose, IBM UK
- Kevin Lynch, Lotus Cambridge
- Robert Merenyi, Lotus Cambridge
- Michele Pennell, Lotus Cambridge
- Glen Salmon, Lotus Cambridge
- Lauren Wendel, Lotus Cambridge

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address: redbook@vnet.ibm.com.

Your comments are important to us!

Part 1
Notes Application Development

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

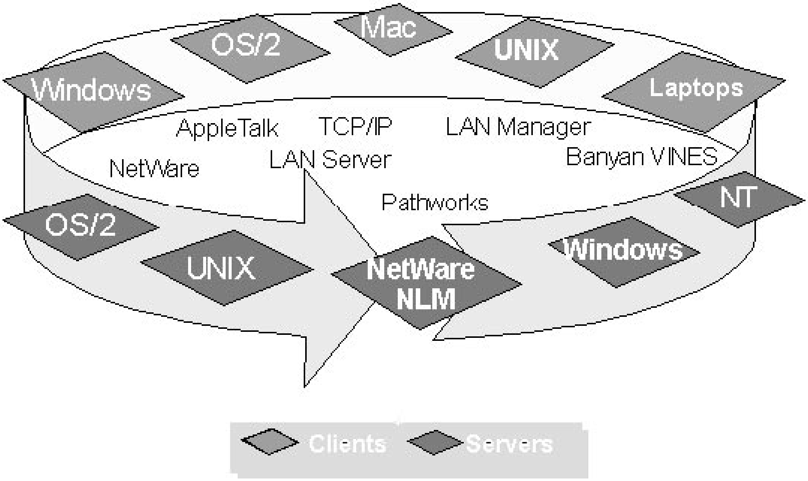
Notes supports the following Network Operating Systems (NOS):

- 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 @DBColumn 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*

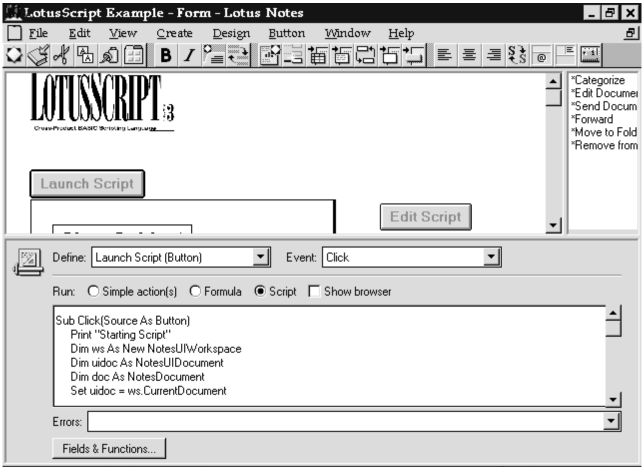
  A forms routing application would focus on sending a form for a specific action. For example, a travel authorization application would send the travel form to the appropriate manager for electronic approval.

- *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:



Define: Launch Script (Button) Event: Click

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

```
Sub Click(Source As Button)
    Print "Starting Script"
    Dim ws As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Dim doc As NotesDocument
    Set uidoc = ws.CurrentDocument
```

Errors:

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

  QueryOpenDocument

  QueryRecalc

  QueryAddToFolder

  QueryPaste

  PostPaste

  QueryDragDrop

  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 2
Creating Notes Databases

This chapter describes how to create and manage Notes databases which are the basic entities of Notes application development. The design elements that a database is made up of are summarized in this chapter and expanded on in the following chapters.

Managing Your Workspace

The Notes workspace is made up of several pages or folders where the Notes databases are displayed as icons. For each page of the workspace, you can specify a title and a color.

As a default, the last tab is assigned to the Replicator, which allows you to manage replication while working away from the office. You cannot rename the Replicator tab and you cannot delete that page.

One of the features of Notes Release 4 is its sensitivity to context. You are very often just one mouse-click away from the properties of the object you are working on. That is true for all design elements: Fields, Buttons, Attachments, Forms, Columns, Views, and Databases.

Managing Notes Databases

What Is a Notes Database?

A database is a collection of related information stored in a single file. The database can represent a small amount of documents used only by a few people, or it can be an enterprise-wide database containing thousands of documents. The upper limit of a database is four gigabytes of data. Every Notes application uses at least one database. Applications of a more complex nature use several databases. For example, a workflow application may route information between databases on one or more servers.

Creating a Database

There are different ways of creating a database. You can:

- Use an existing template
- Use an existing database
- Create a new database

Once the database is created, you can still modify most of the settings using the Database InfoBox. The InfoBox gives you access to more options that we will cover later in this chapter in the section “Changing the Database Properties.”

Using an Existing Template

Notes Release 4 provides a series of written applications that can be used or customized for your own needs. Although there are many types of popular application templates, they are designed to reveal the underlying technology and development capabilities within Notes Release 4. Their main intent is not to be “out-of-the-box” applications.

If your application is identical with, or similar to, an existing template provided with Notes, the most convenient way to create a new database is to use the template as your starting point. Most of the design work has already been done for you. The design elements of the individual templates can easily be copied and pasted into your custom applications.

Listing Available Templates

To see the list of available templates:

1. Choose File - Database - New. The list box on the New Database dialog box lists several templates.

   Tip The shortcut is **CTRL** and **N**.

2. Click the Show advanced templates check box. The list box at the bottom of the list displays additional templates. The templates listed are stored on your local workstation.
3. Select any template you are interested in.
4. Click the About push button to display the database help document. It summarizes what the database can be used for.

To see additional templates stored on a server:

5. Click the Template Server push button.
6. In the Server field, select the server you want to access. Additional templates are listed.

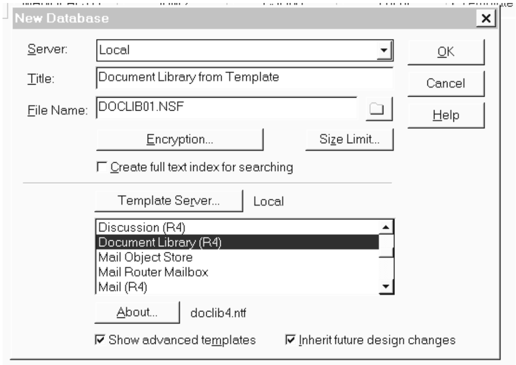
Creating the Database

Follow these steps to create the database:

1. Decide if the database will reside on your local workstation or on a server so that it can be used at once by several users.
2. In the Title field, specify a meaningful title.
3. In the File Name field, specify a file name for the database. You can also take the file name that Notes provides automatically based on the database title.

   Note The extension of a database file is NSF. The extension of a database template file is NTF. The database file extension cannot be changed once the database has been created. It must be unique on the workstation or the server where it is created.

   The following figure gives you an example of a completed New Database dialog box:



New Database

Server: Local

Title: Document Library from Template

File Name: DOCLIB01.NSF

☐ Create full text index for searching

Template Server... Local

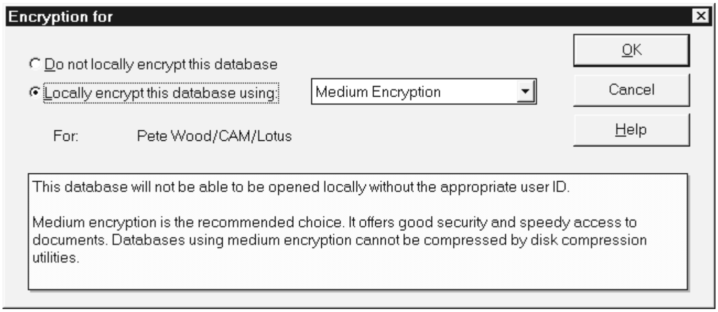
Discussion (R4)
Document Library (R4)
Mail Object Store
Mail Router Mailbox
Mail (R4)

About... doclib4.ntf

☑ Show advanced templates ☑ Inherit future design changes

4. If the database is local, you can encrypt it. This is useful if the database contains confidential data, or if your users have laptops that they will take outside their business locations.

   To specify encryption, click the Encryption push button. Specify the appropriate level of encryption. The following figure shows what you can do:



Encryption for

○ Do not locally encrypt this database

● Locally encrypt this database using: Medium Encryption

For: Pete Wood/CAM/Lotus

This database will not be able to be opened locally without the appropriate user ID.

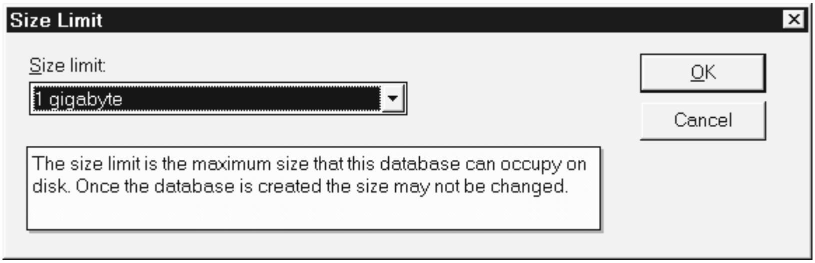
Medium encryption is the recommended choice. It offers good security and speedy access to documents. Databases using medium encryption cannot be compressed by disk compression utilities.

5. If you want to keep the database within a predefined size, click the Size Limit push button found on the New Database dialog box, and select the appropriate size.

   Note The default size is one gigabyte.

   Notes will warn you or the administrator (if the database is on the server) when the size of the database gets close to the specified limit. Make sure that the database size you specify is correct as you will not be able to change this value later on.

   The Size Limit box looks like this:



Size limit: 1 gigabyte

The size limit is the maximum size that this database can occupy on disk. Once the database is created the size may not be changed.

6. If you want your new database design to stay synchronized with the design template, click the Inherit future design changes check box on the New Database dialog box.

Copying an Existing Database

Copying a database is similar to starting from a template, except that you will almost certainly want to change part of the design.

Listing Available Databases

To list the available databases:

1. Choose File - Database - Open.

   Tip The shortcut is **CTRL** and **O**.

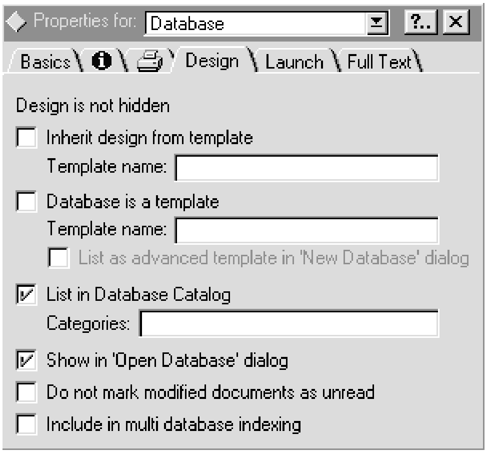
2. If required, specify the appropriate server name in the Server field to list additional databases.
3. Click the About push button to browse the Help document of the database.
4. Click the Add Icon push button to add the database to your workspace.

Creating the Database

To create the database:

1. On your workspace, select the database icon you want to copy.
2. Display the database pop-up menu by clicking the right mouse button.

3. Choose Database Properties.
4. Choose the Design tab. Make sure that the InfoBox shows that the design is not hidden. It should look like this:



Properties for: Database

Design is not hidden

□ Inherit design from template

Template name:

□ Database is a template

Template name:

□ List as advanced template in 'New Database' dialog

☑ List in Database Catalog

Categories:

☑ Show in 'Open Database' dialog

□ Do not mark modified documents as unread

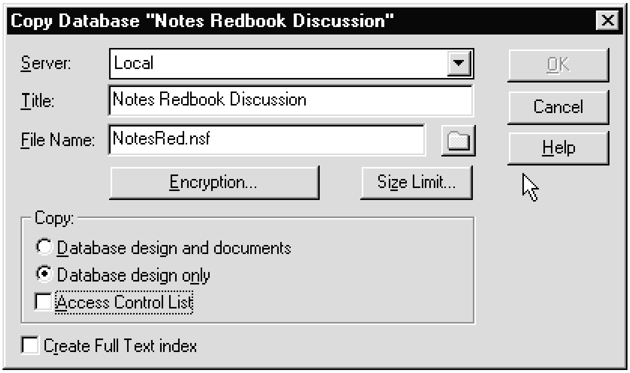
□ Include in multi database indexing

5. Close the InfoBox.
6. Keep the database selected.
7. Choose File - Database - New Copy.
8. Select Local as the server name if you want to store the database on your local workstation. Select a server name if you want to store the database on a server so that several people can access the database.
9. Type a title for the database.
10. Type a file name with extension NSF for the new database.

    Note The file name cannot be changed through Notes once the database is created. It must be unique on the workstation or server where the database is created.
11. Click the Database design only option button since you do not want to copy the documents that are stored in the database.
12. Deselect the Access Control List check box because it could prevent you from modifying the database design in the future.

    Note The access you have to the copy of the database depends on the access you have to the original database.

The following figure gives an example of the Copy Database dialog box:



Copy Database "Notes Redbook Discussion"

Server: Local

Title: Notes Redbook Discussion

File Name: NotesRed.nsf

Copy:

○ Database design and documents

● Database design only

□ Access Control List

□ Create Full Text index

You can optionally select the following two features:

13. Optionally, you can select to encrypt the database if it is a local database. This is useful if the database contains confidential data, or if your users have laptops that they use in public environments.

    To encrypt the database, click the Encryption push button and select the appropriate encryption level.

14. Optionally, you can predefine the database size. Click Size Limit and specify the appropriate size.

    Note The default size is one gigabyte. Notes will inform you or the administrator (if the database is on the server) when the size of the database gets close to that limit. Make sure that the database size you specify is correct as you will not be able to change this value later on.

Creating a New Database

If no template or existing database meets your requirements, you can create a completely new database. This means that you will have to create all the design elements, such as forms, views, and fields. However, you can always copy existing elements from other databases and paste them into the new database.

1. Choose File - Database - New.
2. Type a title in the Title field.
3. From the list of available databases displayed at the bottom of the window, choose the Blank option.
4. Click OK. The new database has been added to your workspace. The database view is displayed. You are ready to start the design of the database.

For more details on how to design a database, refer to the chapters covering Forms and Views respectively.

Changing the Database Properties

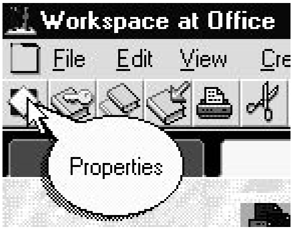
The following describes how you can change the properties of a database.

Displaying the Database Properties

To display the database InfoBox:

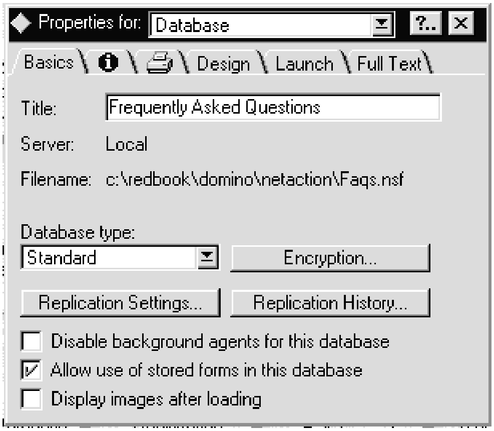
1. Display the database pop-up menu by clicking the right mouse button.
2. Choose Database Properties.

   Tip You can also click the following InfoBox SmartIcon to display the database properties InfoBox.



Specifying the Database Type, Replication, and Encryption

The Basics tab contains information on the database, such as its title, name, location, the replication settings and replication history. The Basics tab looks like this:



Title: Frequently Asked Questions

Server: Local

Filename: c:\redbook\domino\netaction\Faqs.nsf

Database type: Standard

□ Disable background agents for this database

☑ Allow use of stored forms in this database

□ Display images after loading

1. To set the Database type, you can select one of the following values:
   - Standard: Notes database used most of the time.
   - Library: A database with type Library is generally created from the Database library template. It is used to record and store information about the databases located on a Notes server or on a workstation.

If it is located on the server, it contains a list of public databases. This provides an easy way for users to browse the list of databases available to them.

- Personal Journal: It allows you to store personal information. This type of database contains local personal information. It has limited design elements and is meant for individual use.
- Address Book: Creates a database based on the Notes Name and Address Book format.
- Multi DB Search: Is used to specify a database type of Search Through Multiple Databases which uses the SRCHSITE.NTF template. This type of database is used to configure searches among databases that have been designated to participate in Multi Database indexing by selecting the appropriate option on the Design tab of the database InfoBox.

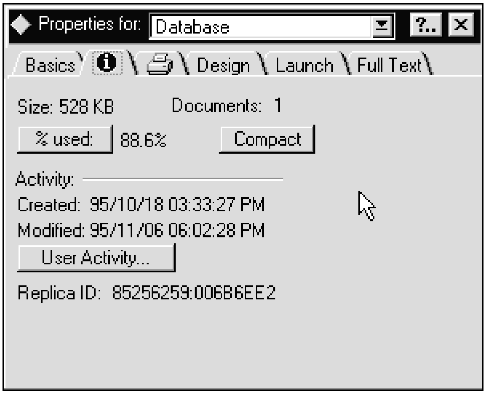
2. Click the Encryption push button to display a window that enables you to specify encryption for the local version of the database.

Displaying General Database Information

1. Click the Information tab to display some general information, such as the size of the database and the number of documents stored.

2. Click the User Activity push button to display information related to user activity.

The Information tab of the InfoBox looks like this:



Properties for: Database

Size: 528 KB Documents: 1

% used: 88.6%

Activity:

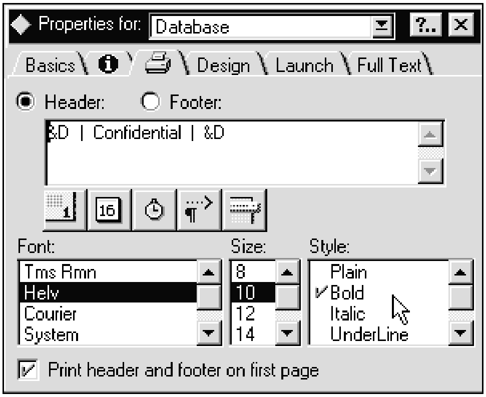
Created: 95/10/18 03:33:27 PM

Modified: 95/11/06 06:02:28 PM

Replica ID: 85256259:006B6EE2

Specifying Print Options

1. Click the printer tab to specify options related to printing the document.
2. Use the icons listed under the Header and Footer option buttons to define the date and time, tabs and page numbering.



Properties for: Database

● Header: ○ Footer:

&D | Confidential | &D

Font: Helv

Size: 10

Style: ✓Bold

☑ Print header and footer on first page

3. You can also select the font, size, and style.

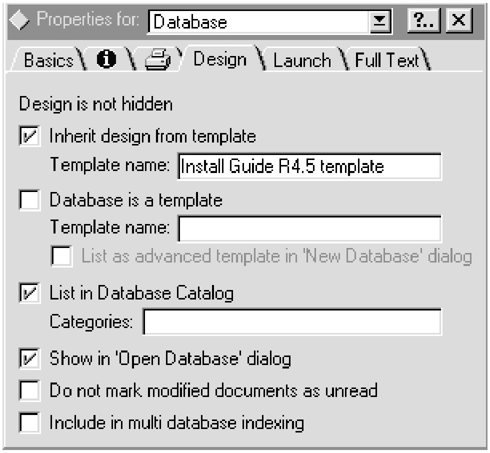
Specifying Database Design Properties

1. Click the Design tab to display or specify information concerning the design of the database.

   The example displayed in the following figure shows that the design of this database is not hidden. Also, the database automatically inherits all the changes made to the template if the template that this database is based on is modified in the future.

2. If the database you are creating is a template, check the Database is a template check box.
3. Specify a name for the template.
4. If appropriate, select the new template to be listed as an advanced template. This indicates that the template should only be customized by Notes developers.
5. Deselect the List in Database Catalog and Show in Open Database dialog check boxes if the database is located on a server, contains sensitive data, and you do not want users to be able to list its name.
6. Select Include in multi database indexing if you want the index to be included in Multi Database Search Database site queries.

The following figure gives an example of the Design tab:



Design is not hidden

☑ Inherit design from template

Template name: Install Guide R4.5 template

□ Database is a template

Template name:

□ List as advanced template in 'New Database' dialog

☑ List in Database Catalog

Categories:

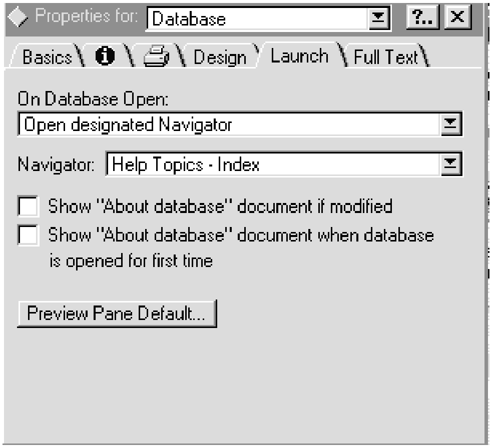
☑ Show in 'Open Database' dialog

□ Do not mark modified documents as unread

□ Include in multi database indexing

Specifying Launch Options

1. Click the Launch tab to define what users will see when they first open the database. The dialog box looks different depending on your choice of action in the On Database Open drop-down list.



On Database Open: Open designated Navigator

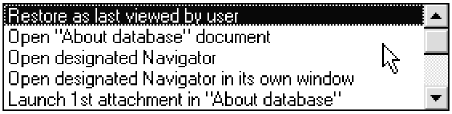
Navigator: Help Topics - Index

□ Show "About database" document if modified

□ Show "About database" document when database is opened for first time

Preview Pane Default...

2. Specify an option from the On Database Open drop-down list. A wide variety of options is possible, such as:

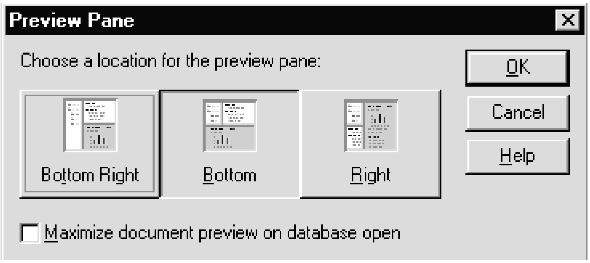


Restore as last viewed by user
Open "About database" document
Open designated Navigator
Open designated Navigator in its own window
Launch 1st attachment in "About database"

3. If you choose one of the two Navigator options, you need to select a navigator from the list of navigators available for the database.

4. Choose the Open designated Navigator in its own window option if you want the navigator to be displayed in a full screen. You would typically choose this option if the navigator consists of a large map or a workflow sketch.

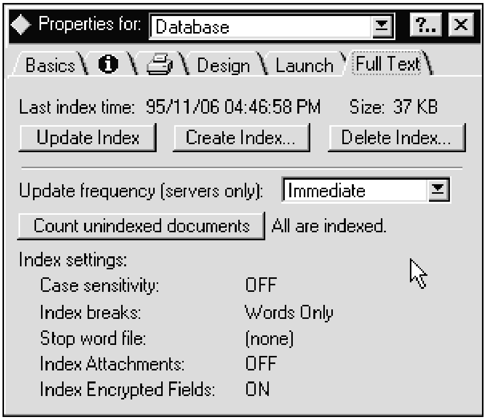
5. You can specify the properties of the preview pane by clicking on the Preview Pane Default button. You are presented with the following choices. Click on the most appropriate for the user.


Preview Pane
Choose a location for the preview pane:
Bottom Right
Bottom
Right
OK
Cancel
Help
□ Maximize document preview on database open

Specifying Full-Text Indexing

1. Display the Full Text tab to create, update, or delete a full-text index, which allows for a fast retrieval of documents.

2. Specify the update frequency as required.



Last index time: 95/11/06 04:46:58 PM Size: 37 KB

Update frequency (servers only): Immediate

All are indexed.

Index settings:

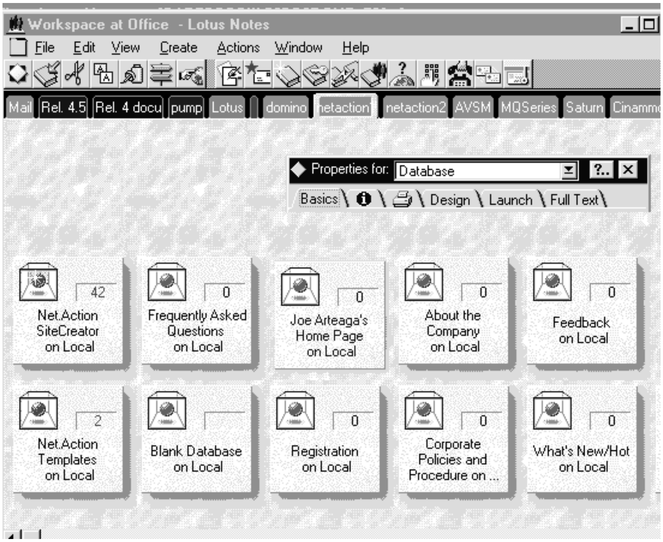
Case sensitivity:	OFF
Index breaks:	Words Only
Stop word file:	(none)
Index Attachments:	OFF
Index Encrypted Fields:	ON

Minimizing the InfoBox

You can keep the InfoBox open while you are working on the design of the database. You will notice that the contents of the InfoBox is refreshed while you are progressing with the design.

You can also reduce the size of the InfoBox by double-clicking on its title bar. This allows you to keep the InfoBox open. Double-clicking on the title bar or clicking on one of the tabs will restore it to its full size.

In the following example, the InfoBox is minimized and located in the upper right-hand corner of the Notes workspace.



Basics on Database Building Blocks

Every Notes application includes at least one database. Each Notes database contains three basic components: forms, fields and views.

In addition, icons, help documents, Navigators, agents (previously known as Notes macros), sections, actions, formulas, and scripts play an important role in giving an application sophisticated automation and processing power.

For each design element of Notes, you need to define the action to be performed. You can generally do this using the @Functions or LotusScript programs to build functional flow between the user and the design elements.

The following gives an overview of the basic elements that developers design for a Notes database application.

Designing Forms

Notes developers create forms to provide the user with a skeleton to create and retrieve information. Forms contain several design elements, such as fields to store text and image data, and buttons to process the form data or to access related data located within or outside of Notes. In each database, there is at least one form but there are generally more, each serving a particular purpose.

All the user data is stored in the Notes database as documents. To create and store a document in the database, the user chooses a form from the Create pull-down menu, fills in the form, and saves it as a document in the database.

Note Forms are mandatory in databases.

Designing Subforms

A subform is an object within a form that can be reused across forms within the database. A subform can contain a logo, a set of fields, or buttons. A form can contain several subforms.

Note Subforms are optional in forms.

Designing Collapsible Sections

You can create sections within a form that can be expanded or collapsed depending on the user's need to view that particular piece of information. For example, you can create a form containing an explanatory section, which can be collapsed after it has been read. You can also assign controlled access to collapsible sections.

Note Sections are optional in forms.

Designing Navigators

Navigators are a new type of view. They represent a graphical "table of contents" of a database. You can configure navigators to present information in a graphic fashion that opens views and folders, or provides links to other navigators. You can also configure navigators to drop documents into folders and run @Functions or LotusScripts, making it possible to easily implement more sophisticated applications such as workflow and messaging.

Note Navigators are optional in databases.

Designing Fields

A field is an element in a form that contains data stored in a specific format.

There are several types of fields, such as editable and computed fields. They support different formats including:

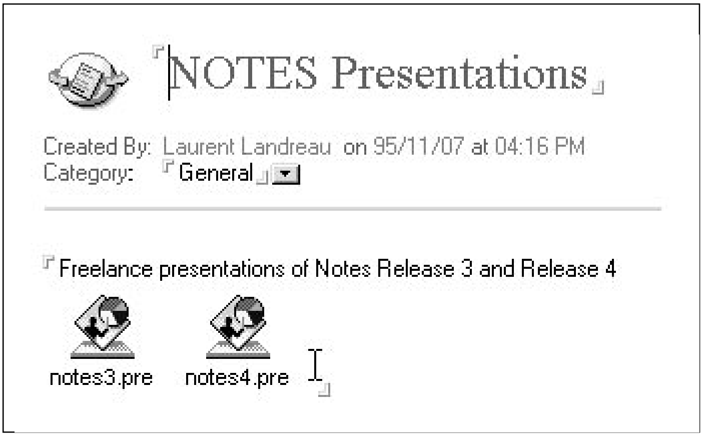
- Text
- Rich text, for large amounts of text, graphics, attached files or multimedia objects
- Number
- Date and time
- User names or group names
- Keywords that can be displayed as option buttons, check boxes or lists

Fields can also be:

- Shared to be reused in several forms on an individual basis or grouped in subforms that allow the reuse of a whole set
- Stored in access-controlled sections to restrict access to the information to predefined users

The following figure shows a form taken from the Document Library template. The form contains fields that can be edited (NOTES Presentations), and fields that are computed (the Created By field).

The Category field provides a list from which the user can select one or several entries.



NOTES Presentations

Created By: Laurent Landreau on 95/11/07 at 04:16 PM

Category: General

Freelance presentations of Notes Release 3 and Release 4

In addition, two Freelance presentations are stored in an editable, rich-text field. Clicking on either icon will start Lotus Freelance and open the presentation.

Note Fields are mandatory in forms.

Designing Hotspots

There are several types of hotspots:

- Pop-up text
- Buttons
- Links to databases or views or documents.

Hotspots can be implemented as buttons that allow users to perform simple tasks, for example, shortcuts to Notes menus, or complex tasks linked to the document processing. These tasks can be written using @functions or LotusScript programs.

The links can be created when designing the form to point to a specific database or view, or they can be created dynamically using a set of @functions. They enable you to program links between documents. The user can then open documents using a single mouse-click.

Note Hotspots are optional in forms.

Designing Tables

Tables are useful to summarize information or line up fields in rows and columns. You can define static tables where you predefine in advance the number of rows. Or, you can define dynamic tables where the number of rows can be adjusted to the user's needs.

Note Tables are optional in forms.

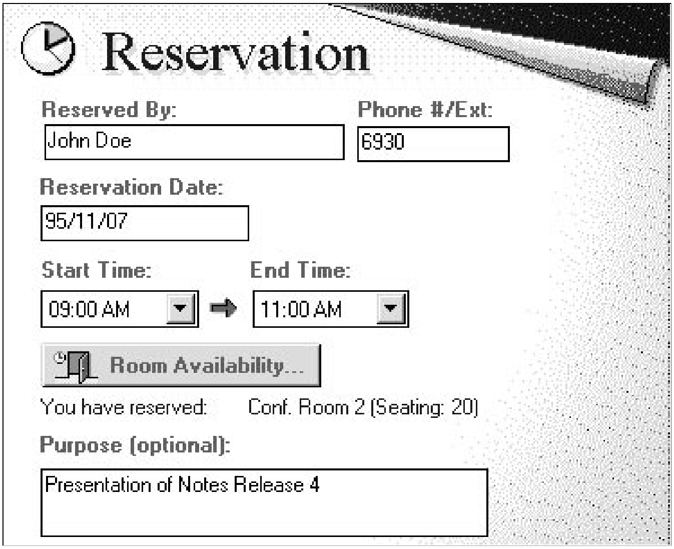
Designing Layout Regions

Layout regions provide you with a graphical interface for laying out text, graphics, fields, and other components when creating a form or subform.

While designing a form, you can insert a layout region frame within the form. Within this frame, you can place text, graphics, and other fields. You can create them within the frame, or you can drag them from other locations within the form. This enables you to use layout regions in forms requiring text-overlaying graphics, where-placed fields, and field-tab sequencing.

Note Layout regions are optional in forms.

The following figure shows an example of a layout region taken from the Room Reservation template. It contains text, buttons and drop-down lists.



Reservation

Reserved By: John Doe

Phone #/Ext: 6930

Reservation Date: 95/11/07

Start Time: 09:00 AM

End Time: 11:00 AM

You have reserved: Conf. Room 2 (Seating: 20)

Purpose (optional): Presentation of Notes Release 4

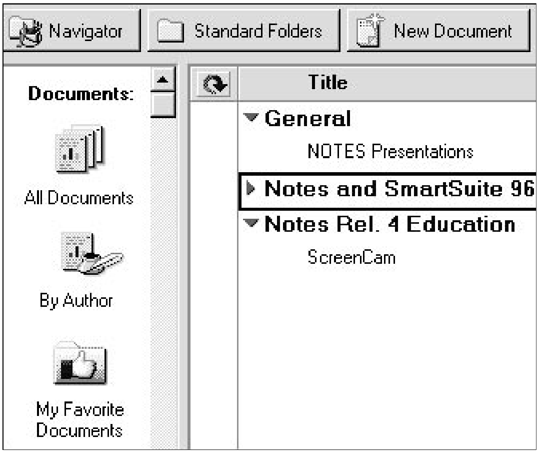
Designing Views

A view displays a summary of the documents contained in a database using a row-and-column format. The view generally lists the key fields of documents but it can also do basic calculations, such as totals and averages.

There can be several views in the database, each one presenting the data in a particular way. For example:

- Different fields can be displayed
- You can display all the documents, or only documents by author, for example
- You can apply different sort criteria for columns to be displayed

The following figure shows a typical Notes Release 4 view taken from the Document Library database:



There is an action bar at the top of the window and a navigator on the left-hand side.

On the right side, the documents are displayed and categorized. The arrows next to the documents (they are also called “twisties”) indicate whether a category is expanded or collapsed, that is, if the documents related to that category are listed or not. To read or edit a document in a database, the user opens a view and selects and opens the document shown as a row in the view.

Note Views are mandatory in databases.

Designing Folders

Users can create folders to organize documents displayed in views in a different way. When users store documents in folders, they actually store pointers to the documents. The documents can be moved from folder to folder, removed from the folder, or deleted from both the view and the folder.

Note Folders are optional in databases.

Designing Action Bars

The action bar is a new feature of Release 4, which you can use in forms, views, and folders. It is a non-scrollable set of buttons displayed under the SmartIcon bar. Each button performs an action that a user would typically do either using the menu bar or using the different buttons provided in the form. In a view, you could, for example, provide an action called “Move to Folder” to ease the user’s work. In a form, you could put the basic actions, such as Save, Close, and Response to Document on the action bar. This would enable users working within a document to quickly perform those actions.

Note Action bars are optional in databases.

Designing Agents

Agents are procedures that a developer or user can write to provide automation for specific actions. Such actions can be, for example:

- Archiving old documents
- Categorizing mail depending on the author of the memo or on the existence of keywords in the subject field
- Periodic creation and mailing of summaries of newly created documents.

Note Agents are optional in databases.

Designing Icons

An icon represents the database on the workspace. It can be drawn or copied from an existing application such as Windows PaintBrush. Icons on the workspace should be unique in their appearance.

Note Icons are recommended in databases.

Writing Help Documents

You should write Help documents to provide the user with at least some basic information regarding the purpose of the database and the usage of database forms and views. You should also document the key fields and actions that can be performed in a form.

Note Help documents are recommended in databases.

In the following chapters, we are going to see examples of each of the design elements described above. We will also look at some of the template databases included with Lotus Notes.

Chapter 3
The Notes Integrated Development Environment

Introduction

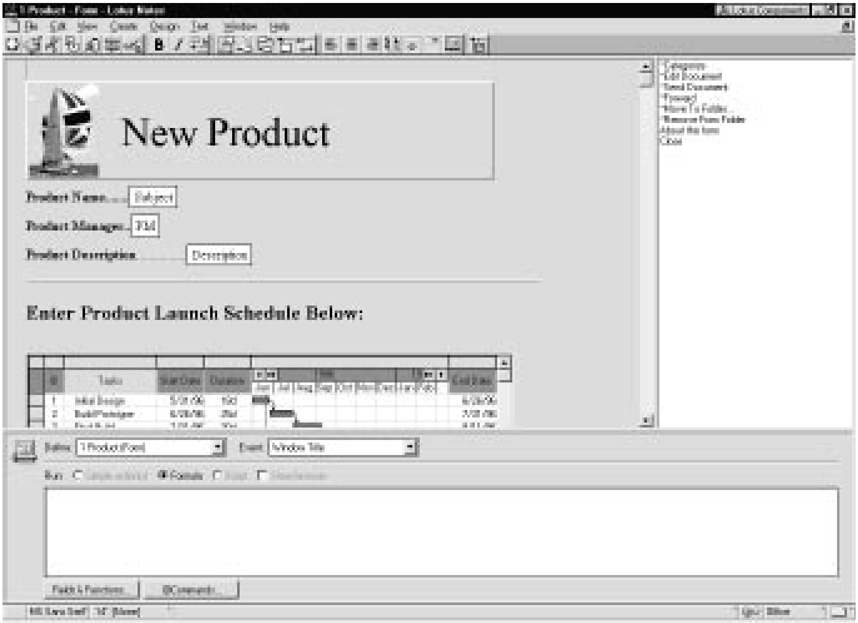
Lotus Notes provides a complete integrated application development environment which allows you to create a variety of elements that are used in a Notes application. A typical Notes application consists of several or all of the following:

- Forms — which provide the templates through which data in the application is entered and displayed. Unlike a traditional template, forms can also act on the data. That is, they may contain application logic and processing. For example, when a user inputs information in the form, the form might, depending on the contents, send an E-mail message to another user.
- Views and folders — which provide different ways of looking at all or part of your data, according to specified criteria. A view might be thought of as similar to a report in a traditional database program, except that the view is dynamic and includes links to information in the application.
- Navigators — which provide graphical means of moving between views.
- Agents — which add functionality to the application. For example, you might create an agent that once a day scans the documents in the database, checks the contents of certain fields, and places documents that meet specified criteria into a special folder.

There are, within Notes, Integrated Development Environments (IDEs) for developing each of the above named elements. These IDEs share many common elements. For example, the tools for writing LotusScript are identical among all of the IDEs. We will focus on the forms IDE since it is the most complex, and therefore the most interesting one. Keep in mind, though, that much of the discussion in this chapter applies to the development of all Notes elements, not just forms.

Elements of the Forms Integrated Development Environment

The picture below illustrates the IDE you use when creating and editing forms in Lotus Notes.



The three components of the Integrated Development Environment are represented in the picture.

Main Design Window

This is the large window at the top left of the screen where you visually design your form. Using this window you can add static text, fields, layout regions, and embedded objects to the form.

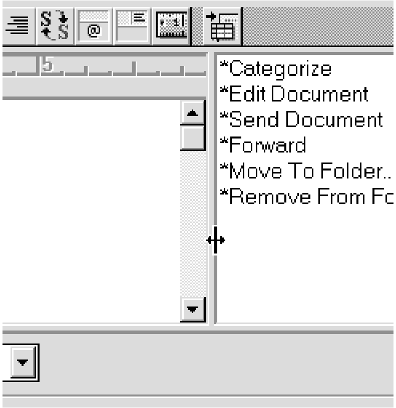
To work in this window you place your cursor at the desired position on the form and type your text, or use the menus or SmartIcons to insert the desired object. When you are working in this window the status bar at the bottom of the screen provides controls you can use to quickly format text.

Action Pane

This is the narrow window at the top right of the screen. The action pane is used to define actions that are associated with the form. An action is a LotusScript procedure, simple action, or macro that performs work when activated. Actions can be invoked from the menu bar, the action bar, or by LotusScript procedures.

The action pane is not visible in the Integrated Development Environment by default. To view the action pane, do one of the following:

1. Choose View - Action Pane, or
2. Drag the right-hand edge of the main design window to the left, as shown in the following figure, until the action pane is the size you want.



*Categorize
*Edit Document
*Send Document
*Forward
*Move To Folder..
*Remove From Fc

To hide the action pane you can either drag its left border all the way to the right, choose View - Action Pane to uncheck the menu option, or click on the appropriate SmartIcon.

Design Pane

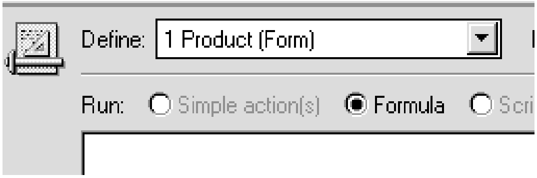
The design pane is the window below the main design window and the action pane. This is where all the programming work in Notes, be it using simple actions, the formula language or LotusScript, takes place.

The design pane is, by default, visible when you create a new form or open one for editing. You can also drag the border between the main design window and the design pane to change the size of both, or to hide the design pane altogether. You show or hide the design pane by choosing View - Design Pane, just as you do to show and hide the action pane.

The design pane is also used when you are creating or editing a database or view action.

There are a number of components to the design pane. Let's look at each one of them.

Define Box

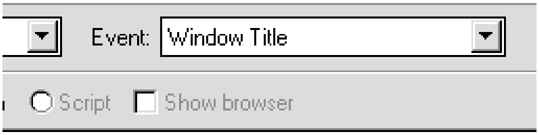


Define: 1 Product (Form)

Run: ○ Simple action(s) ● Formula ○ Scri

The define box is a combo box that shows you all of the objects on your form that can be programmed, along with all of the actions defined for the form.

Event Box

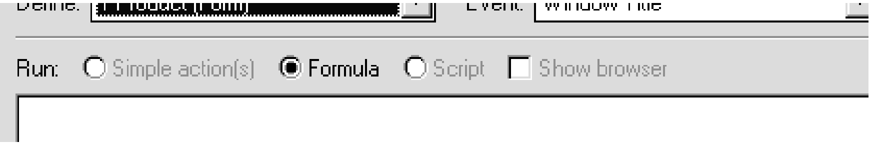


Event: Window Title

○ Script □ Show browser

This combo box shows all of the programmable events for the object showing in the define box. Each object has its own set of events, so the contents of this box will change in accordance with the object specified in the define box. There are also some cases where no events are available, in which case the event box is not shown. Also, LotusScript procedures or |subs| which you add to your Notes application will appear in this list.

Run Area

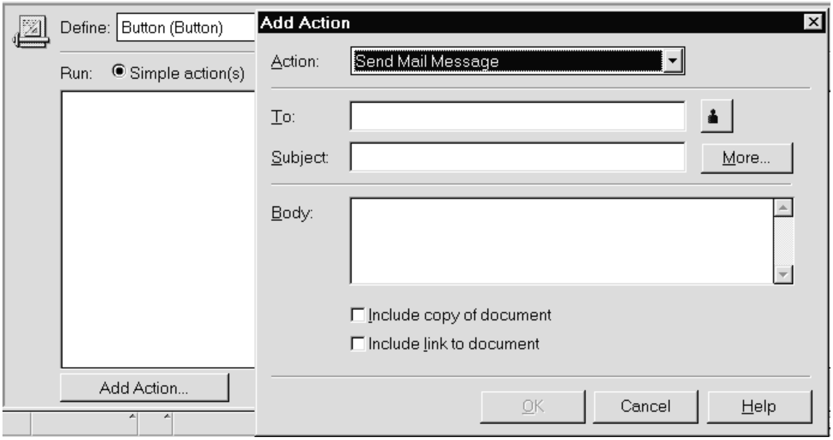


Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

The three option buttons specify the type of programming that you will apply to the specified object and event. Choose one of the following:

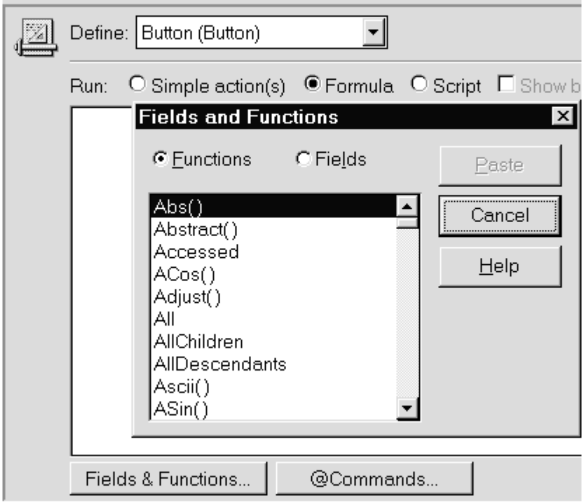
- Simple Action(s): Lets you easily specify one or more actions from a number of pre-defined actions, such as Modify Field, Send Document, Move Document to Folder, etc. When you specify Simple Action(s), an Add Action button (see following figure) appears at the bottom of the design pane. Clicking this button brings up a dialog box which allows you to specify the action you wish to perform. Not all objects on the form support simple actions. If an object which does not support simple actions is selected in the define box, this option button is disabled. To edit an existing simple action, select it with the mouse and then click the button; the button changes to Edit Action when the action is selected.

When you click on the Add Action button, a dialog window to specify a pre-defined action appears as in the following figure.



- Formula: Lets you write Notes formula language macros and commands that will run when the specified event occurs for the object. When you specify Formula, a Fields & Functions button (see following figure) appears at the bottom of the design pane. Clicking this button brings up a dialog box which will display all of the fields defined on the form, or all of the functions available in the Notes formula language. Double-clicking one of these fields or functions inserts it into the editor window at the current cursor position.

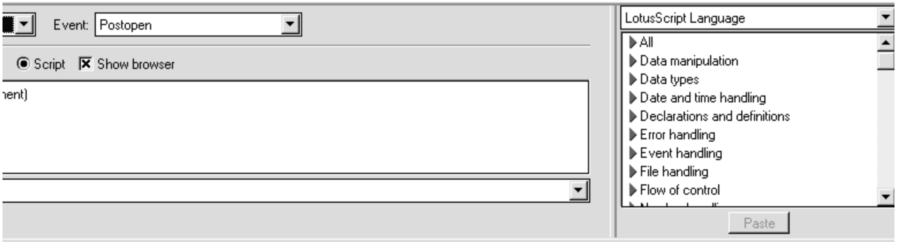
  When you click on the Fields & Functions button, a dialog window to specify a function or a field appears as in the following figure.



When certain events are selected an @Commands button will appear next to the Fields & Functions button. Clicking this button brings up a dialog box which will display all of the @Commands available in the Notes formula language. Not all objects on the form support formulas. If an object which does not support formulas is selected in the define box, this option button is disabled.

- Script: Lets you write LotusScript procedures that will run when the specified event occurs for the object. When this option is selected, the Fields & Functions button and an Error combo box are made available. Not all objects on the form support LotusScript. If an object which does not support LotusScript is selected in the define box, this option button is disabled.

The last control in the run area is the Show Browser check box. This control is only enabled when the Script option button is selected. When this control is checked, the object browser is displayed, as shown below:



Event: Postopen

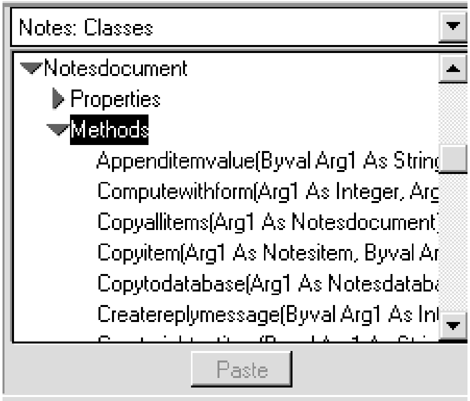
● Script ☒ Show browser

LotusScript Language

The browser is a ready reference of the LotusScript language and functions, and of all of the objects, their properties and methods. For example, if you want to quickly determine all of the methods available for the NotesDocument class, show the browser and then:

1. Select Notes Classes from the top combo box. All the Notes object classes are displayed.
2. Scroll the browser's list box until you find the NotesDocument entry.
3. Click the triangular "twistie" icon to expand the listing under NotesDocument. You will see three entries, for Properties, Methods and for Events. The Events entry has no twistie, meaning there are no events defined for this class.

4. Click the Methods twistie, and a list of all of the NotesDocument methods will be listed, as shown below:



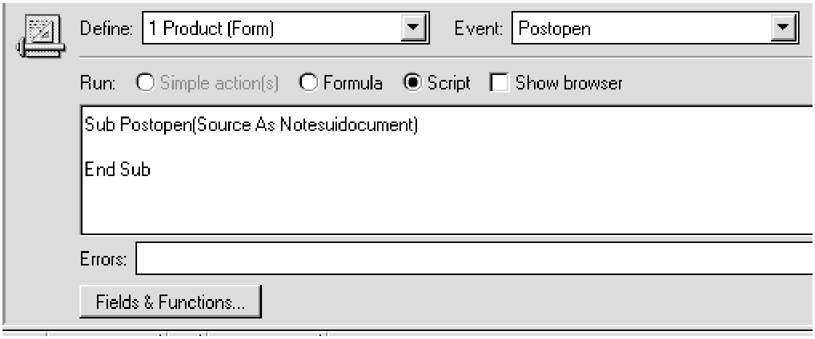
```
Notesdocument
    Properties
    Methods
        Appenditemvalue(Byval Arg1 As Strin
        Computewithform(Arg1 As Integer, Arg
        Copyallitems(Arg1 As Notesdocument)
        Copyitem(Arg1 As Notesitem, Byval Ar
        Copytodatabase(Arg1 As Notesdataba
        Createreplymessage(Byval Arg1 As Int
```

5. Double-click the method or property in the browser to insert the prototype code into your Notes application.

Script Editor and Formula Editor

The larger window below the run area is where you program the actions that Notes will execute. If the Formula option button is selected, this area is the formula editor. If the Script option button is selected, this area is the script editor.

When the Script button is selected, the script editor will automatically enter the appropriate Sub and End Sub statements for the specified object and event, as the following picture illustrates:


Define: 1 Product (Form) Event: Postopen

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

```
Sub Postopen(Source As Notesuidocument)

End Sub
```

Errors:

Error Box

This box (see figure above) is only displayed when the Script button is checked. It lists all of the syntax errors that Notes detected in your LotusScript. If there are multiple errors, clicking the arrow to the right of the error box will expand this combo box so that all syntax errors found are displayed. You can navigate to a specific error in the list, no matter where in the application it is located, by selecting it from the drop-down list.

When a syntax error is corrected, Notes will remove the error indication from the error box. Notes will not allow you to save a form with LotusScript syntax errors, so you will have to fix all errors (which will result in an empty error box) before you will be able to save your form. If you wish to save an application with errors in it, comment out the sections that contain errors, or copy the contents to the clipboard, remove the error, and save the application.

Working With the Script Editor

The script editor functions very much like a text editor. The standard text editor key conventions are used, such as:

- **HOME** places the cursor at the start of the current line.
- **END** places the cursor at the end of the current line.
- **CTRL+HOME** places the cursor at the start of the script.
- **CTRL+RIGHT ARROW** moves the cursor one word to the right.

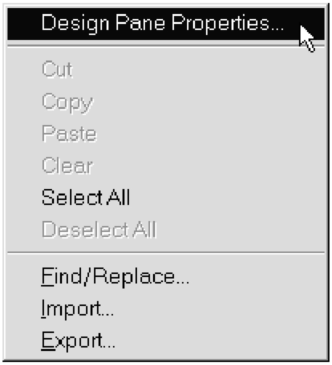
You can select text in the usual way (using the **SHIFT** and arrow keys, or by dragging the mouse pointer over the text to be selected). The clipboard-related menus and SmartIcons, such as Cut and Paste, are available when you are working in the script editor, as well as the corresponding accelerator keys, such at **CTRL+C** and **CTRL+V**. This means that you can cut and paste scripts, or script fragments, from other objects in Notes.

Design Pane Properties

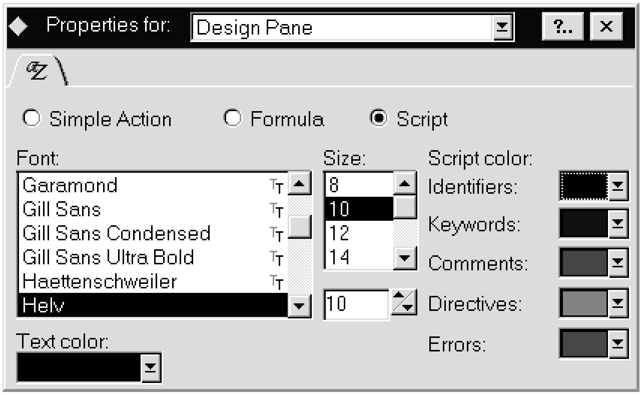
The properties of scripts in the programming pane can be changed as follows:

- Script color
- Font properties

1. Click the right mouse button on the programming pane.
2. Select Design Pane Properties to set the properties of scripts displayed in the programming pane.



The Design Pane InfoBox is displayed:



Properties for: Design Pane

○ Simple Action ○ Formula ● Script

Font:
Garamond
Gill Sans
Gill Sans Condensed
Gill Sans Ultra Bold
Haettenschweiler
Helv

Size:
8
10
12
14

10

Text color:

Script color:
Identifiers:
Keywords:
Comments:
Directives:
Errors:

Setting the Script Color

To set the script color, follow these steps:

1. On the Design Pane InfoBox, select the Script option button.

   The Script color options are only available when you select the Script option button.

   There are 5 types of categories for which the font color can be changed:

   - Identifiers

     You can change strings, such as, variable name, event name, class name.

   - Keywords

     LotusScript keywords, built-in procedures, functions and so on.

- Comments

  Comment lines between %REM and %END REM directives

  Strings starting with a single quote (')

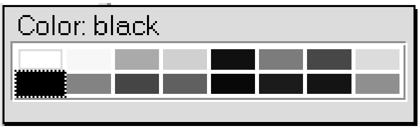
- Directives

  %INCLUDE, %REM and so on

- Errors

  Lines which have syntax errors

2. Drop down the list of available colors and for each script type, choose one of the 16 colors available:



Color: black

Note Once you have set the script color properties, they are in effect in any programming pane (forms, agents, and so on) within the same Notes database.

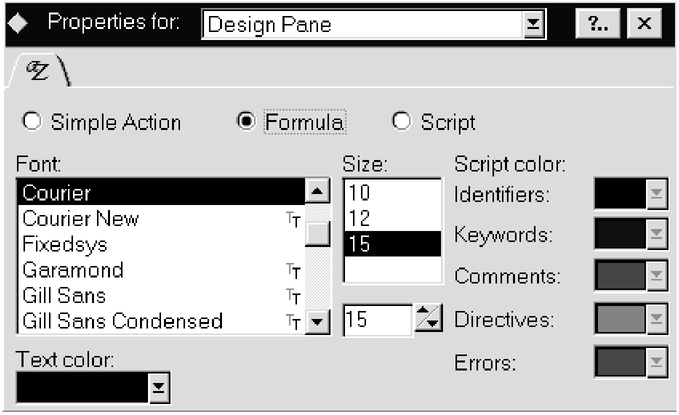
Setting the Font Properties

To set the font properties, follow these steps:

1. On the Design Pane InfoBox, select the Formula option button, for example.
2. Specify the font type, font size, and text color of the formula language shown in the programming pane.

Note You can specify the same features for scripts and simple actions.

Any font you want can be used, but a monospaced font is generally a good choice for program statements.



Properties for: Design Pane

○ Simple Action ● Formula ○ Script

Font: Courier

Courier New
Fixedsys
Garamond
Gill Sans
Gill Sans Condensed

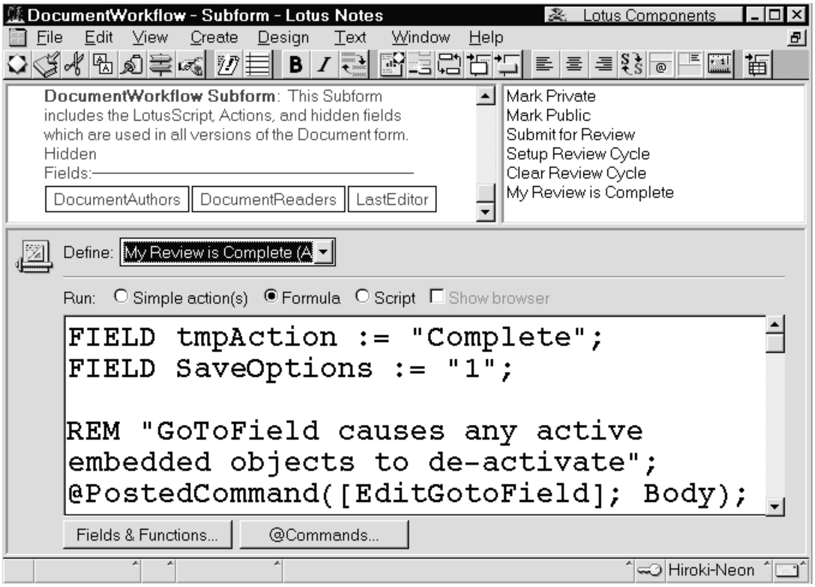
Size: 15

10
12
15

Text color:

Script color:
Identifiers:
Keywords:
Comments:
Directives:
Errors:

Selecting the above properties results in the following figure:


DocumentWorkflow - Subform - Lotus Notes
Lotus Components
File Edit View Create Design Text Window Help

DocumentWorkflow Subform: This Subform includes the LotusScript, Actions, and hidden fields which are used in all versions of the Document form.

Hidden Fields:

Define: My Review is Complete (A

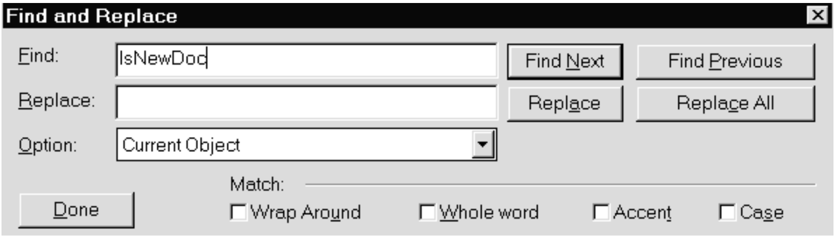
Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

```
FIELD tmpAction := "Complete";
FIELD SaveOptions := "1";

REM "GoToField causes any active
embedded objects to de-activate";
@PostedCommand([EditGotoField]; Body);
```

Searching Your Scripts

You can search all your LotusScript programs in a database as well as the scripts displayed in a programming pane. If desired, you can replace the search string that was found. This feature is only available for LotusScript source programs.

1. Select the Script option button in the programming pane.
2. To display the Find and Replace dialog box, select Edit - Find / Replace or click the right mouse button on the programming pane and then select Find / Replace.



Find: IsNewDoc

Replace:

Option: Current Object

Match: □ Wrap Around □ Whole word □ Accent □ Case

3. If you'd like a case sensitive search, select the Case check box.

4. You can choose a range for the search string.

   The following three options are available:

   - Current Object

     All of the events in an object including the scripts which you are currently working with. For example, all of the events in a Form object.

   - Current Section Only

     Only one event in an object currently displayed. For example, a PostOpen event in a Form object.

   - All Objects

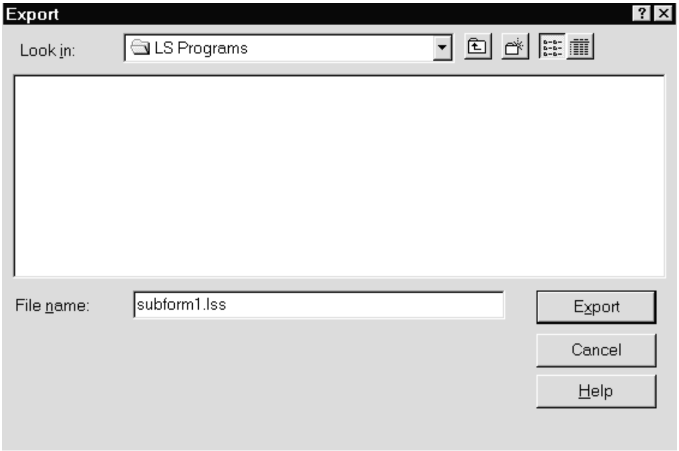
     All events in all the objects of a design component. For example, a form, a navigator and so on.

Exporting Script Programs

When you debug or maintain your script programs in a Notes database, you may want to print them out and archive them. When there are many events which hold large scripts, it is quite hard to copy and paste them into another editor. In that case, you should consider exporting all of your script programs using the export feature. There are three export features available, which are the same as the Find / Replace options.

Note The export feature is only available for LotusScript source programs. If you'd like to get a summary of all the Formula programs in a Notes database, select File - Database - Design Synopsis.

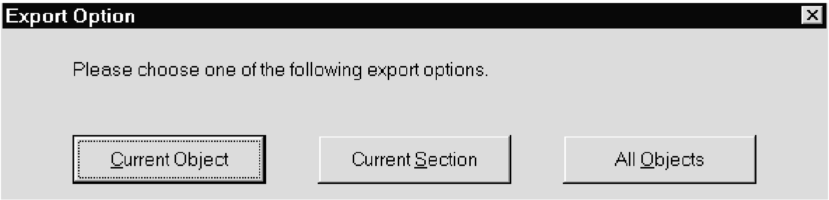
1. Select the Script option button on the programming pane.
2. To display the Export dialog box, select File - Export or click the right mouse button on the programming pane and then select Export.



Look in: LS Programs

File name: subform1.lss

3. Specify a directory where you want to save the scripts and a file name and click the Export button. The following dialog box is displayed:



Please choose one of the following export options.

You can choose one of the following three export options:

- Current Object

  All of the events in an object including the scripts which you are currently working with. For example, all of the events in a Form object.

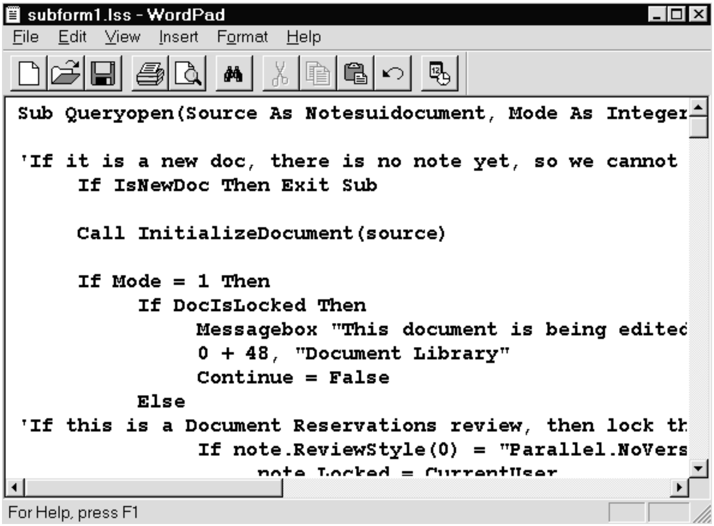
- Current Section Only

  Only one event in an object currently displayed. For example, a PostOpen event in a Form object.

- All Objects

  All events in all the objects of a design component. For example, a form, a navigator and so on.

The following figure shows an example of script programs exported from a Notes database.



```
Sub Queryopen(Source As Notesuidocument, Mode As Integer

'If it is a new doc, there is no note yet, so we cannot
     If IsNewDoc Then Exit Sub

     Call InitializeDocument(source)

     If Mode = 1 Then
          If DocIsLocked Then
               Messagebox "This document is being edited
               0 + 48, "Document Library"
               Continue = False
          Else
'If this is a Document Reservations review, then lock th
               If note.ReviewStyle(0) = "Parallel.NoVers
                    note.Locked = CurrentUser
```

Once you have modified the scripts, you can import them into a Notes database as described in the following section.

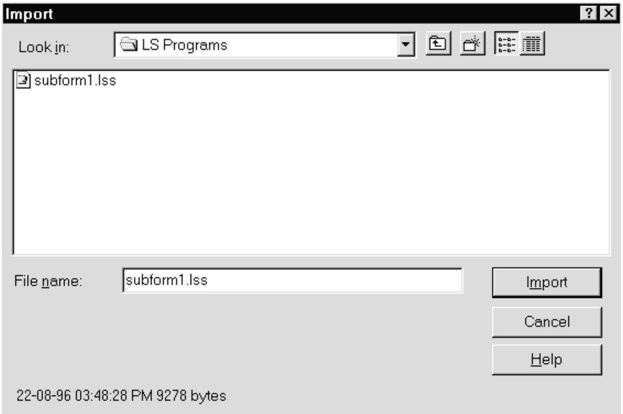
Importing Script Programs

When you create script programs, you may want to use your favorite editor instead of the Notes script editor. There are two choices for you to use your script programs created outside the Notes IDE. One is to use the %Include statement to include your script file (it should be a plain text file) into a Notes application.

Note Scripts included by %Include cannot be debugged using the LotusScript debugger. They must be imported bug-free.

The other choice is to use the Import feature described here. It allows you to import a script file into a programming pane. This means that scripts are imported into a current object. This feature is only available for LotusScript source programs. The Export option All Objects is not available for importing scripts.

1. Select the Script option button on the programming pane.
2. To display the Import dialog box, select File - Import or click the right mouse button on the programming pane and then select Import.



Look in: LS Programs

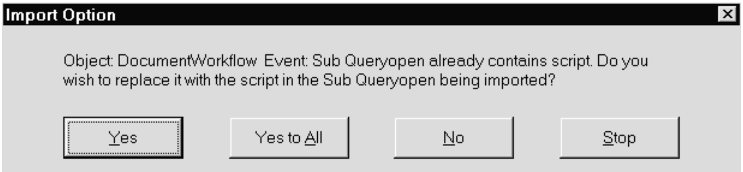
subform1.lss

File name: subform1.lss

22-08-96 03:48:28 PM 9278 bytes

3. Specify a directory where to save the scripts, and a file name.
4. Click the Import button.

Note If you try to import scripts which have been exported using the Export option All Objects, you may see the following dialog box. Select an appropriate action as required by your program environment. Be aware that your script may become corrupted if you choose a wrong action.



Object: DocumentWorkflow Event: Sub Queryopen already contains script. Do you wish to replace it with the script in the Sub Queryopen being imported?

Special Script Editor Features

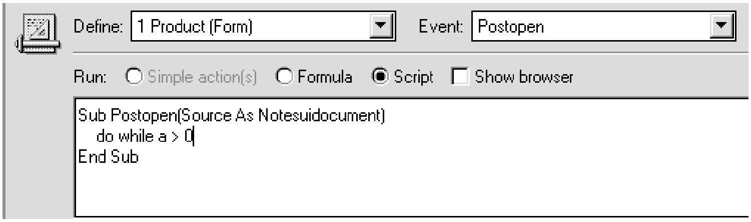
One feature already mentioned is the capability of looking up LotusScript functions and objects in the browser. If you click on any entry in the browser, that entry becomes highlighted, and you can click the Paste button at the bottom of the browser to paste that line at the current cursor position in the script editor. You can also double-click the entry to copy it to the script editor.

Each time you press the **ENTER** key, or move off a line of LotusScript, the script editor checks that line for syntax errors, and also capitalizes the LotusScript reserved words, that is, the words in the statement that are a component of the LotusScript basic language.

When you enter structured programming statements such as For, While, Do, Select Case, etc., the script editor automatically does the following:

- Inserts the corresponding ending statement (for example, Loop for the Do statement) below the statement you typed.
- Inserts a blank line between the two statements, with the cursor being placed on that line so you can continue to type your code.
- Automatically indents the statements within the construct.

The following figure shows the state of the script editor after you type the opening statement of a Do loop:

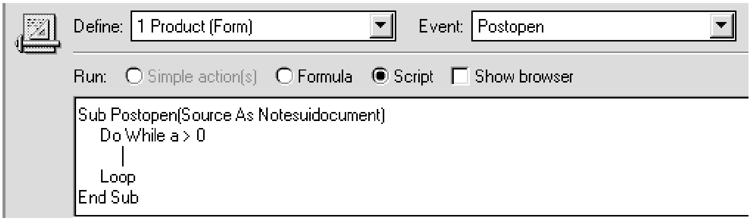


Define: 1 Product (Form) Event: Postopen

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

```
Sub Postopen(Source As Notesuidocument)
   do while a > 0
End Sub
```

When the **ENTER** key is pressed, the script editor window changes as shown in the following figure:



Define: 1 Product (Form) Event: Postopen

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

```
Sub Postopen(Source As Notesuidocument)
   Do While a > 0

   Loop
End Sub
```

Chapter 4
Designing Application Forms

When you have completed this chapter, you should know:

- How to build and modify forms
- How to change the form attributes
- How to create fields in a form, and change their attributes
- How to use various design elements, such as subforms, buttons, layout regions, and the action bar
- How to create program DocLinks

Using the Document Library Template

The Document Library template is an ideal database to start with. This is where you can find the largest number of the features that make Notes Release 4 so impressive.

In addition to the @Functions and the interface that Release 3 developers already know, the Document Library template contains a lot of the Release 4 features. Here are some of them:

- Interface
  - Action Bar
  - Navigator
  - Layout Region
  - SubForms
- Programmability
  - Simple Action
  - LotusScript
  - Agents
  - Workflow and document routing

You will see some more as we go along.

Notes is an excellent container to store documents. The Document Library database itself is an electronic filing cabinet that stores reference documents. The documents are sent to reviewers for comments.

To create your working database:

1. Choose File - Database - New.
2. In the Title field, type a title for the database, for example, My Document Library.
3. In the File Name field, type a file name for the database, for example, MYDOCLB.NSF.
4. Select the Document Library (R4) entry from the template list.
5. Make sure you remove the check mark next to Inherit future design changes as we are going to do some modifications to the design. We do not want them to be overridden by changes made on the server version of the database.
6. Click on the OK button. The database is created on your workspace and the first document of the database is displayed. This is the help document "About the Document Library."
7. Leave the document by pressing the ESC key, or by choosing File - Close. The My Document Library - All Documents view is displayed. It is currently empty.

Creating Sample Documents

Create two sample documents to get a better impression of what the database looks like. To do so, perform the following steps:

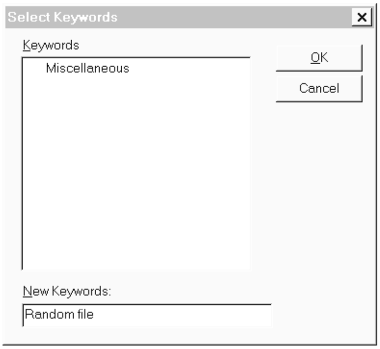
1. Click the New Document push button on the action bar.

   Tip You can also choose Create - Document.

   The New Document form is displayed. You can see several kinds of fields:

   - The cursor is on an entry field of format Text
   - Three computed fields in the Created By line: Author of the document, of format Author Name, and two fields of format time, one containing the date, the other containing the time
   - Next to the static text showing the word Category, an entry field of format keyword list
   - An entry field, of format rich text
2. Specify the data of the first document. In the cursor field, type, for example, My first document.
3. Click on the arrow next to Category.
4. Deselect Miscellaneous in the Keywords list.

5. In the New Keywords field, add your own keyword, for example, Random file. The Select Keywords window looks like this:



New Keywords: Random file

6. Click OK. The new keyword is displayed in the Category field.
7. Use the tab key to go to the last field.
8. Choose File - Attach and select any file from the list displayed.
9. Click the Create button. An icon is pasted into the field.
10. Click the Close button on the Action Bar and confirm that you want to save the document.
11. Create a second document.
12. Specify the data of the second document. In the cursor field, type, for example, My second document.
13. Click on the arrow next to Category.
14. Leave the Miscellaneous option selected.
15. Click the Mark Private push button on the Action Bar.
16. Click the Close button of the Action Bar and save the document. Now that you are back in the view, you can see:
    - Next to each category is a twistie, an arrow that points downwards or horizontally when clicked. It shows if a category is expanded or collapsed.

      Note This arrow replaces the following formula used in Release 3 to expand or collapse a list:

      ```
      @IsExpandable(“+”; “”)
      ```

      In Release 4, no code is necessary any more!
    - Next to the second document, a lock icon indicates that the document is private.

      Note Notes supports 170 different icons in view columns.

Chapter 4 covers twisties and icons in more detail.

Using Forms

The form is the skeleton you provide to users. They can enter data either by typing or by using buttons. There is always at least one form in a database. Most often though, business applications have more than one form, each form being targeted to the type of information the user wants to save in the database.

The form contains all the design elements: fields to store the user's information, and static text, buttons, sections, and subforms that help the user gain access to the information.

To create a new form, choose Create - Design - Form. Or, you can copy and paste a form from the Design Form pane and from there perform your customization.

In this chapter, you are going to edit the properties of an existing form, the Response form, to give you an idea of what is available to you.

Specifying Form Properties

The Form InfoBox box contains all the information related to forms.

To look at the form properties, do the following:

1. From the standard navigator expand the Design option.
2. Choose Forms.
3. Double-click on the Response form listed in the view pane to open up in design mode.
4. Click the Properties SmartIcon.
5. In the InfoBox displayed, click on the *Properties for:* combo box and select Form. An InfoBox box is displayed that allows you to set the properties of the form. It consists of six tabs:
   - Basics
   - Defaults
   - Launch
   - Background
   - Print
   - Security

Using the Basics Tab

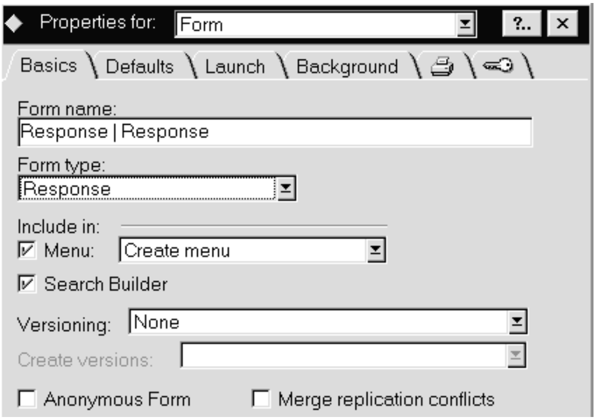
The Basics tab stores general information about the form.

1. In the Form name field, specify a name and an alias for the form.

   By default, the form name appears as an item in the Create option on the menu bar. It is the name the user sees. Therefore, make it as meaningful as possible.

2. If desired, add an underscore next to one of the letters of the name to be used as an accelerator key.

3. It is recommended that you create an alias for each name. This is the name you will use in your code. Specifying an alias enables you to leave your code unmodified if the user requests to have the name of the form changed, for example.



Form name: Response | Response

Form type: Response

Include in:

☑ Menu: Create menu

☑ Search Builder

Versioning: None

Create versions:

□ Anonymous Form □ Merge replication conflicts

4. Specify the form type. This form is a response type, which means that it is linked to a main document of type document, and that it cannot exist without this parent. A third type is the Response to Response type, which adds a third level to the document hierarchy.

5. If you want to include the form in the Create option on the menu bar, you can select:

   - Create menu if there is only a small number of forms. Up to nine forms are possible in the Create option.
   - Create - Other dialog if there are many different forms. It is recommended to put the least used forms with the Create - Other dialog option.

6. Click the Search Builder check box to add the response form to the list of forms that users can search.

7. In the Versioning field, specify whether or not you want version control. These options are possible:
   - None
   - New versions become responses
   - Prior versions become responses
   - New versions become siblings
8. You can decide to keep track of the different versions of the document that have been created. The current version can be set as a response to the previous version of the document, or vice versa.
9. Check the Anonymous Form check box if you want authors or editors to anonymously put documents based on this form into the database.

   Note Documents created with an anonymous form do not contain the $UpdatedBy field but have a $Anonymous field with a value of 1.

   Of course, you need to make sure that the author's or editor's name does not appear in any other field of the form.
10. If desired, check the Merge replication conflicts check box. Notes then merges conflicting edits into a single document whenever possible. If two users edit different fields in the same document, Notes saves the changes to each field in a single document. The field replication feature new to Release 4 makes this possible.

    However, if two users edit the same field in the same document, Notes saves one document as a main document and the other as a response.

Using the Defaults Tab

The Defaults tab lets you specify details regarding the usage of the form.

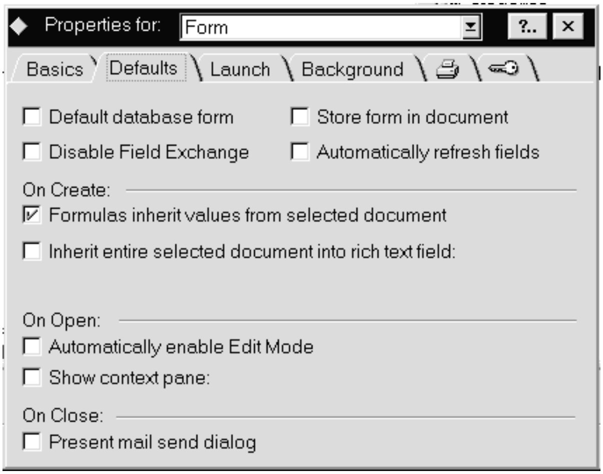
1. Leave the Default database form check box deselected as our current form is a response form.

   Notes uses a default form to open documents whenever their associated forms have been dropped from the database design. You should select this option for the main form of the database.
2. Leave the Store form in document check box deselected.

   You must store the form into the document if a user that has no access to the database, for example, if a user receives a document and has no access to the design of the form used to create the document.

   Note Selecting the Store form in document option increases the amount of disk storage required to store each document based on the form.
3. Leave the Disable Field Exchange check box deselected to enable data exchange with Notes/FX compliant applications.

4. Leave Automatically refresh fields deselected. If it is critical to users that fields be recalculated as they fill in the form, you must select this option. But for performance reasons, it is better to refresh selected fields through formulas or scripts when the user clicks on a button or saves the document.



☐ Default database form ☐ Store form in document

☐ Disable Field Exchange ☐ Automatically refresh fields

On Create:

☑ Formulas inherit values from selected document

☐ Inherit entire selected document into rich text field:

On Open:

☐ Automatically enable Edit Mode

☐ Show context pane:

On Close:

☐ Present mail send dialog

5. In the On Create section, leave the first check box selected. This ensures that the data contained in the fields of the parent document are copied to the fields of the response document.

   In the Document Library template, the subject of the main document is copied to the OriginalSubject field of the response form. Here is the formula of the default value for the OriginalSubject field:

```
@If(@IsAvailable(OriginalSubject);
    OriginalSubject;
    Subject)
```

   The subject of the main document is copied into the OriginalSubject field when the response document is created.

6. If desired, check the Inherit entire selected document into rich text field option, and specify how the fields of the parent document will be displayed in the response document. There are three possible options:

   - Link
   - Collapsible rich text
   - Rich text

7. If desired, in the On Open section, check the Automatically enable Edit Mode check box. This opens the document in edit mode.

   Note The default is browse mode.

8. Check the Show context pane check box if you want Notes to divide the window into two parts:

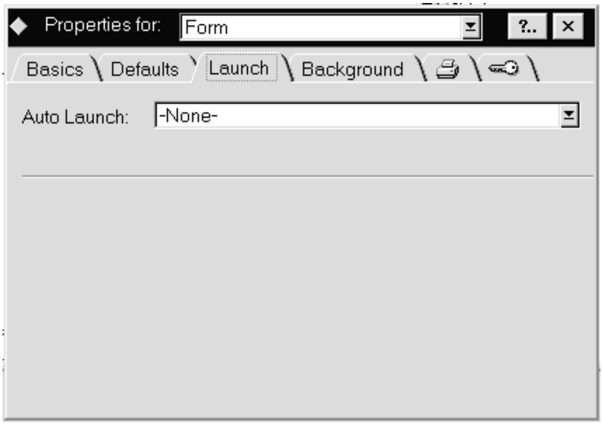
   - The Response document you open is in the top half of the window
   - The Parent or linked document is in the bottom half

9. If desired, check the Present mail send dialog check box. This ensures that the user is presented with the Mail window, but can only be used if you have created a SendTo field in the form, which contains the addressee of the document.

Experiment with the Defaults page at your leisure. Before leaving it, make sure that the Inherit entire selected document and the Show context pane check boxes are deselected.

Using the Launch Tab

The Launch tab enables you to specify what happens when the document is opened.



Properties for: Form

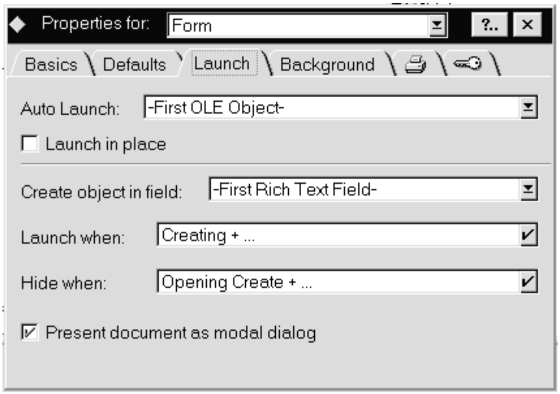
Auto Launch: -None-

1. In the Auto Launch field, specify the type of action to take place when the document is opened. These are the available options:

   - None
   - First Attachment
   - First Document Link
   - First OLE Object

2. Specify First Attachment if the database contains different types of attached documents, such as ASCII files, presentations, or word processor files.

3. If you choose First OLE Object, the InfoBox looks like this:



Properties for: Form

Auto Launch: -First OLE Object-

☐ Launch in place

Create object in field: -First Rich Text Field-

Launch when: Creating + ...

Hide when: Opening Create + ...

☑ Present document as modal dialog

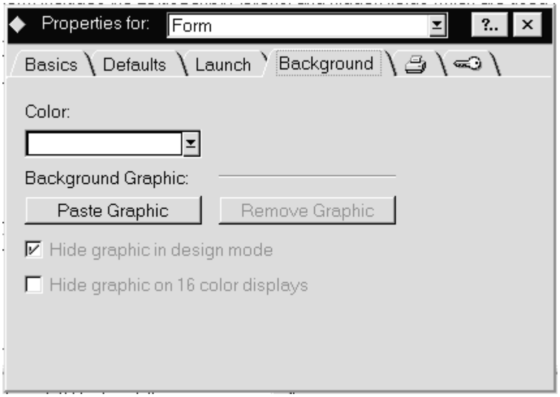
4. Specify additional information as required.
5. Before you leave the Launch tab, make sure that None is selected in the Auto Launch field.

Using the Background Tab

1. Click the background tab to specify the options about the background on the form.
2. Specify the background color for the form using the Color drop-down combo box.
3. If desired, click the Paste Graphic button to paste a graphic image into the form. If the image is smaller than the form, Notes tiles the image to conform to the size of the form.

   Note You need to copy an image to the clipboard before you click the button.

   Tip Keep in mind that the cursor could be difficult to see on some displays if you choose a color such a yellow, for example.



Properties for: Form

Color:

Background Graphic:

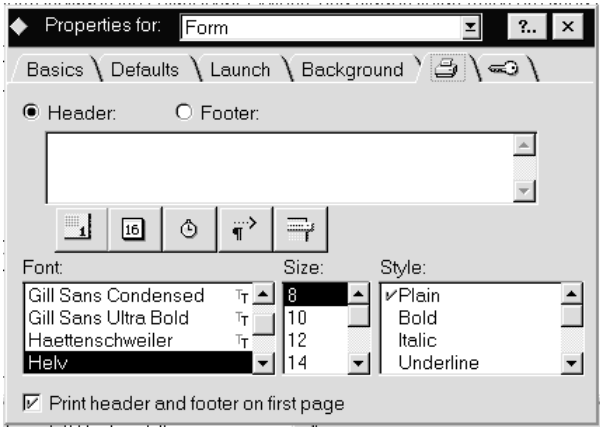
☑ Hide graphic in design mode

☐ Hide graphic on 16 color displays

Using the Print Tab

1. Click the print tab to specify options related to printing a document based on the form.

2. Use the icons listed under the Header and Footer option buttons to define the date and time, tabs, and page numbering.



Properties for: Form

● Header: ○ Footer:

Font:
Gill Sans Condensed
Gill Sans Ultra Bold
Haettenschweiler
Helv

Size:
8
10
12
14

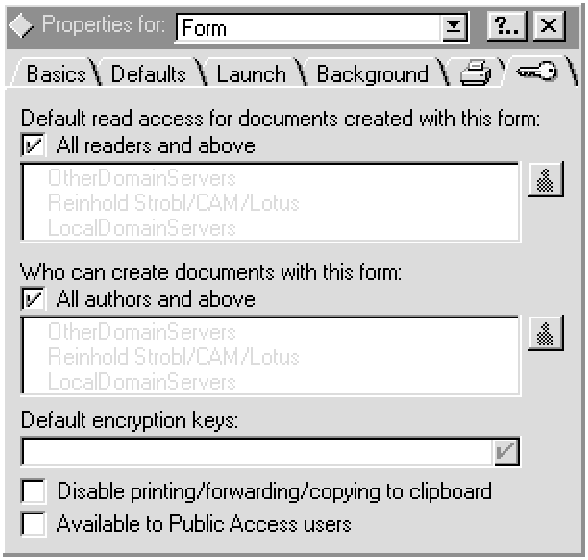
Style:
✓Plain
Bold
Italic
Underline

☑ Print header and footer on first page

3. You can also select the font, size, and style.

Using the Security Tab

Use the security tab to define which users or user groups are authorized to use the form.



Properties for: Form

Default read access for documents created with this form:

☑ All readers and above

OtherDomainServers
Reinhold Strobl/CAM/Lotus
LocalDomainServers

Who can create documents with this form:

☑ All authors and above

OtherDomainServers
Reinhold Strobl/CAM/Lotus
LocalDomainServers

Default encryption keys:

□ Disable printing/forwarding/copying to clipboard

□ Available to Public Access users

1. Deselect the All readers and above check box. This activates the small button 👤 to the right of the list.
2. Click the button. A window is displayed that allows you to select the users and groups from the different address books you have access to.
3. Specify who can create documents with this form. The default is all authors and above. If required, deselect the check box and click the small button to the right of the list. A window is displayed that allows you to select users and groups.
4. If required, select disable printing/forwarding/copying to clipboard. This makes it more difficult for users to distribute the documents created with this form to other users. However, it is recommended to limit this option to confidential data.

   Note Selecting this option does not prevent the usage of other software to copy data to the clipboard.
5. Select Available to Public Access users, if required.

Giving the Form a Title

You can define actions to be performed when users trigger events as they compose, edit, or browse a document created with the form. The events can be defined in the programming pane at the bottom of the Form Builder window.

One event you can define is the window title. When you design a form, make sure you give it a title. The title will then appear on the Notes title bar when a document is created or edited based on the form.

1. Close the Form InfoBox and return to the Form Builder window.
2. Take a look at the formula used to set the window title of the response form:

```
DEFAULT OriginalSubject :=
@If(@IsAvailable(OriginalSubject); OriginalSubject;
Subject);
NewRespTitle := "New Response to \"" + OriginalSubject +
"\"";
StandardTitle := "Response " + @DocNumber("") + " of " +
@DocSiblings + " to \"" + OriginalSubject + "\"";
@If(@IsNewDoc; NewRespTitle; StandardTitle)
```

OriginalSubject is a field of the form. *NewRespTitle* and *StandardTitle* are work variables whose scope is limited to this formula.

The variable *NewRespTitle* stores the New Response text string concatenated with the variable *OriginalSubject* that contains the subject of the parent document.

The variable *StandardTitle* stores the concatenation of the sequence number of the response (@DocNumber), the number of responses linked to the parent document (@DocSiblings) and the subject of the parent document.

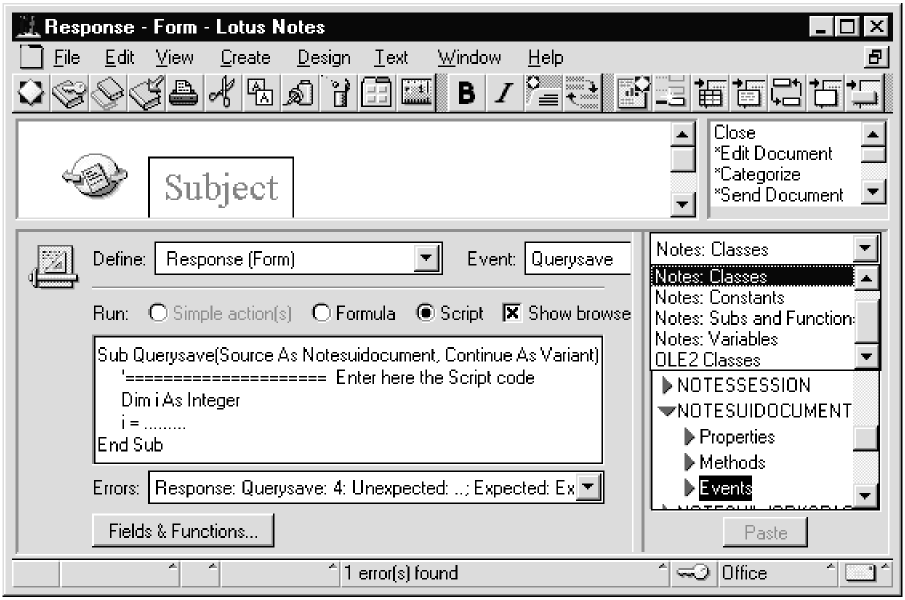
If the document is a new document (@IsNewDoc), the title is set to *NewRespTitle*. If the document already exists in the database, the title is set to *StandardTitle*.

Looking at Form Events

In the programming pane, you can define additional actions. You can code processes to be performed:

- When the document is initialized
- Before or after the document is opened (Queryopen and Postopen)
- Before or after the document changes its mode from browse to edit or from edit to browse (Querymodechange and Postmodechange)
- When the contents of document fields have been recalculated (Postrecalc)
- When the document is saved (Querysave) or closed (Queryclose)
- When the document terminates

Here is an example of the programming pane when you code the process Querysave:



Define: Response (Form)

Event: Querysave

Run: ○ Simple action(s) ○ Formula ● Script ☒ Show browse

```
Sub Querysave(Source As Notesuidocument, Continue As Variant)
    '====================  Enter here the Script code
    Dim i As Integer
    i = .........
End Sub
```

Errors: Response: Querysave: 4: Unexpected: ..; Expected: Ex

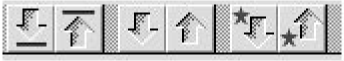
You can see that:

- The code you write is LotusScript as the Script option button is selected.
- The browser is displayed on the right. It gives you access to Notes and OLE2 classes, the LotusScript syntax, constants, variables, and other things.
- The Querysave function provides two parameters. One of them is an instance variable of the Notesuidocument class. It is the unique ID that identifies the document being saved. In the browser, you see that you have access to all the properties, methods and events of that class.
- The LotusScript compiler has found one syntax error. It is listed in the Errors combo box showing the source line number.
- The Fields & Functions push button gives you access to all fields of the forms, even to the @Functions as it is possible to imbed @Functions into LotusScript.

Creating a New Field

We are going to look at a fast way to go from the child document created with the Response form to its related parent.

Of course, you could use the following Navigation SmartIcons to go from one document to the other.

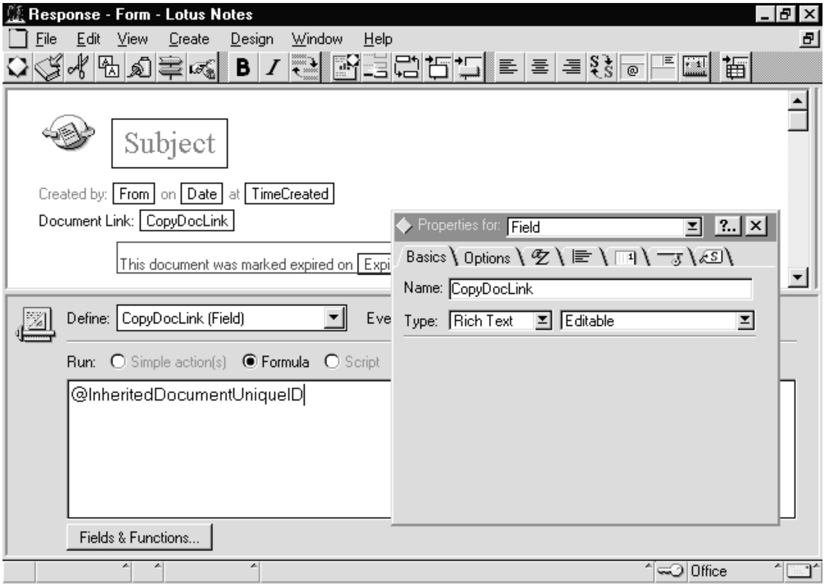


However, this can be cumbersome when you need to scroll up a number of child documents before you reach the related parent.

To provide a more elegant solution, we are now going to create a field in the Response form. It will be used to store a reference, or DocLink, to the parent document. This will enable you to quickly access the parent document when browsing the response document. It could be especially useful if you want to refer to some of the information you are responding to.

1. You should still see the Form Builder window of the response form displayed. If not, from the standard navigator, choose Design, then Forms, and open the Response form listed in the view pane.

2. Type the static text as shown in the following example. The static text is Document Link.


Response - Form - Lotus Notes
File Edit View Create Design Window Help
B I
Eve
Basics Options
Fields & Functions...
Office

Subject

Created by: From on Date at TimeCreated

Document Link: CopyDocLink

This document was marked expired on Expi

Properties for: Field

Name: CopyDocLink

Type: Rich Text Editable

Define: CopyDocLink (Field)

Run: ○ Simple action(s) ● Formula ○ Script

```
@InheritedDocumentUniqueID
```

3. Next to the static text, create the field CopyDocLink. To do so, choose Create - Field.

   Tip You can also display a pop-up menu using right-click and choose Create Field.

4. On the InfoBox for Field, type a name for the field, for example, CopyDocLink. The new name is now also shown in the Define field in the programming pane.

5. In the Type field, choose Rich Text.

6. In the programming pane, type the following:

   ```
   @InheritedDocumentUniqueID
   ```

7. Leave Default Value as the event type.

8. Save the modified form by pressing the **ESC** key, and confirm that you want to save the form.

9. Close the information box that is displayed.

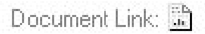
Performing a Test Run

To test your modification:

1. From the standard navigator, select All documents.

2. In the view pane, select one of the documents you created earlier, and click the Response button on the Action Bar.

3. In the response document, you see that a document link has been created. It looks like this:

   Document Link:

4. Double-click on the small icon. The parent document is displayed in full screen. It overlaps the response document.
5. Click the Close button on the Action Bar to close the parent document.
6. You are now back in the Response document. Type Response 1 in the cursor field.
7. Click the Close button on the Action Bar and save the document.

In the list of documents, the response document now appears indented under its parent document.

Sharing and Reusing a Field

It can also be useful to create a DocLink field in the Response to Response form. As the same logic is applied, we can reuse the same design as for the DocLink field in the Response form. The DocLink created in the Response to Response document will then point to its parent Response document.

Rather than create a new DocLink field, we are going to share and reuse the field CopyDocLink you just created in the Response Form.

Sharing a Field

To share the field CopyDocLink, perform the following steps:

1. From the standard navigator, choose Design, then Forms. The list of forms is displayed in the view pane.
2. Double-click the Response form. The form is displayed.
3. Click on the field CopyDocLink.
4. Choose Design - Share This Field from the menu bar.
5. Press the ESC key and save the form.

   The field is now shared, and you can reuse it in other forms.

Reusing the Shared Field

You can only reuse fields in the database where the field has been defined as a shared field.

1. From the standard navigator, choose Design, then Forms. The list of forms is displayed in the view pane.
2. Double-click the Response to Response form. The form is displayed.
3. Position the cursor in the same place as you did before in the Response document, and type the static text: Document Link:

4. Next to the static text, right-click to display the pop-up menu of the form.
5. Choose Insert Shared Field. A window is displayed.
6. CopyDocLink should already be highlighted. Click OK. The field is now added to the form.
7. Press the ESC key and save the form.

Performing a Test Run

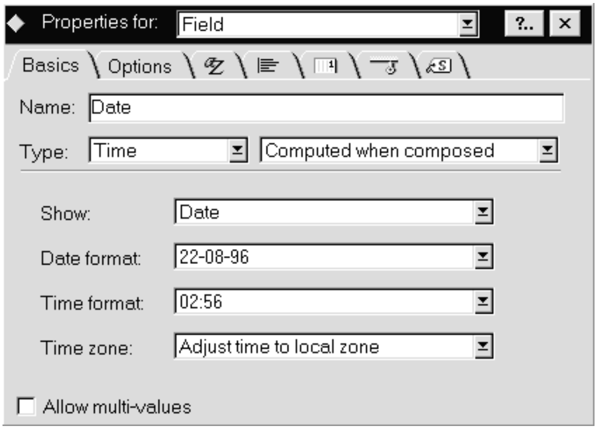
1. On the view pane, choose All documents shown in the left pane.
2. In the document pane, select the Response 1 document you have created earlier.
3. Click the Response to Response button on the Action Bar.
4. In the response document, you see a document link. Double-click it. The Response 1 document is displayed.
5. Double-click the DocLink shown in the Response 1 document. The parent document is displayed.
6. Leave the documents by pressing the ESC key.

Looking at Field Properties

Now that you have created a field, we are going to look at some of the properties of fields contained in the Document form.

Looking at the Date Field

1. From the standard navigator, choose Design, then Forms.
2. Double-click the Document form in the view pane to open it.
3. Double-click the Date field. The Field InfoBox is displayed. It looks like this:



Properties for: Field

Name: Date

Type: Time | Computed when composed

Show: Date

Date format: 22-08-96

Time format: 02:56

Time zone: Adjust time to local zone

□ Allow multi-values

On the Basics tab, Notes displays the field format. Part of it is the Time format. It comprises both Date and Time, each of them supporting several formats.

The Basics tab also shows how the data is actually put into the field. The following types of field are available:

- Editable: The user enters the data, or the data is created when the user selects a button performing a formula or script written by the developer.
- Computed: The field is computed each time the document is created, edited and saved.
- Computed when composed: The field is only computed when the document is created. This type of field is especially useful for storing the name of the author of the document, the creation date or a document reference number.
- Computed for display: The field is computed each time the document is opened in browse or edit mode. The contents of the field are only visible while the document is open. It is not saved into the database and is not visible in a view.

  For example, this type of field is used to display the current time and date or work variables, such as the server name where the database is stored.

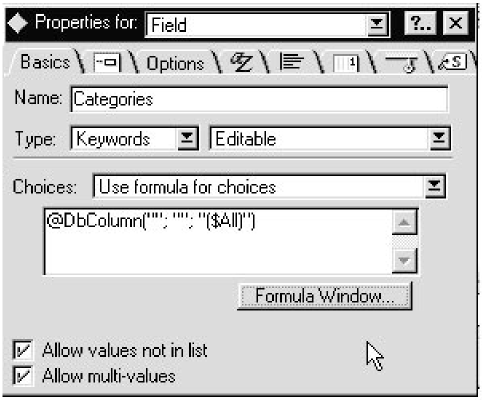
4. Close the InfoBox.

Looking at the Categories Field

Next, look at the Categories field.

The Basics Tab

1. Double-click the Categories field. The InfoBox for this field is displayed, as shown in this figure:



Properties for: Field

Name: Categories

Type: Keywords | Editable

Choices: Use formula for choices

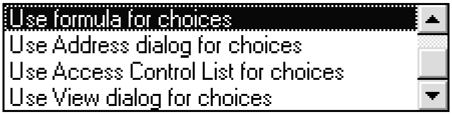
@DbColumn(""; ""; "($All)")

☑ Allow values not in list

☑ Allow multi-values

The field is of type Keywords. There are different ways of displaying the list of keywords from which users can make their selections. In our example, an @DbColumn formula checks all documents in the current database for predefined categories and retrieves them for display in a keyword list.

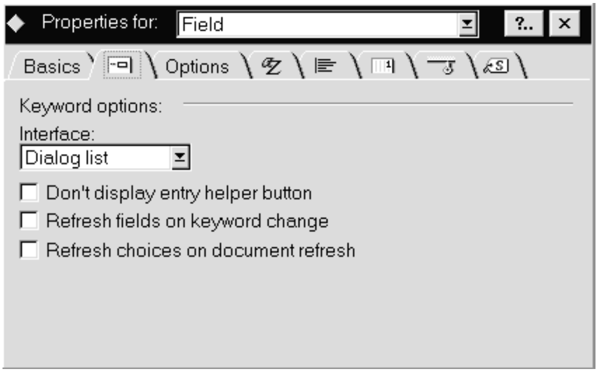
2. Click on the Choices combo box. Here are some of the other choices available:



Use formula for choices
Use Address dialog for choices
Use Access Control List for choices
Use View dialog for choices

Keyword Options

1. Go to tab 2 of the InfoBox. It looks like this:



Properties for: Field

Keyword options:

Interface: Dialog list

□ Don't display entry helper button
□ Refresh fields on keyword change
□ Refresh choices on document refresh

2. In the Interface section, you can specify how the keywords will be displayed. Three options are available:

- Dialog List

  The keywords are displayed in a dialog list. If you want to enable users to specify additional keywords not listed, check the Allow values not in list checkbox on the Basics page of the InfoBox.

- Checkbox

  This is a multiple choice field. If you want the check boxes displayed vertically, leave 1 in the Columns field. If you want the check boxes displayed as a table, set the number of columns to a value greater than 1.

  To display all check boxes on one line, set the number of columns to the number of check boxes available.

- Option Button

  Only one option button can be selected at any given time. The option buttons can be displayed vertically, horizontally or as a table.

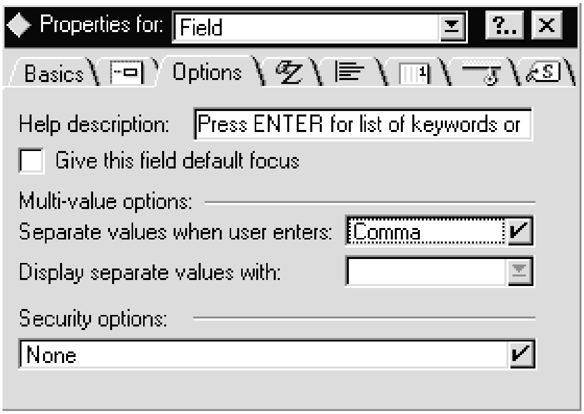
Next, take a look at the programming pane:

The initial value has been set to “Miscellaneous.” The Event combo box enables you to define other actions:

- Input Translation: It can be used to modify the data entered by the user, to trim blanks, and to change users’ names into uppercase or propercase
- Input Validation: This requires a choice from a list, which is done using an @If formula. The same validation can be written using LotusScript. To use LotusScript, you need to choose the Exiting option in the Event combo box. The Script option button is automatically selected.

Options Tab

- Go to tab 3 of the InfoBox. It looks like this:



Properties for: Field

Help description: Press ENTER for list of keywords or

□ Give this field default focus

Multi-value options:

Separate values when user enters: Comma

Display separate values with:

Security options:

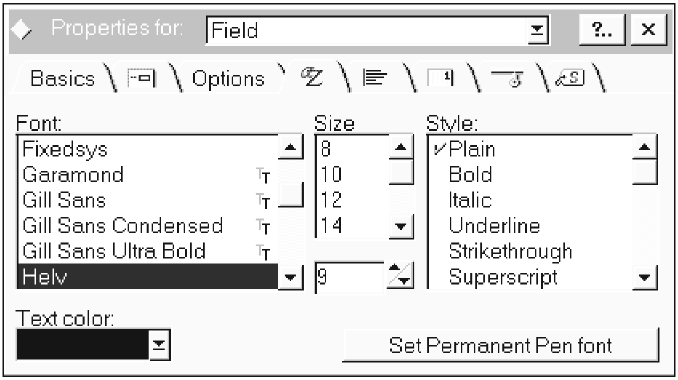
None

This tab enables you to specify:

- Field help
- Whether the entry field will have the initial focus when the form is opened. You must specify this option if you want to place the cursor in an entry field that is not the first one in the form. You can also use this option if you want to paste data in a particular entry field before placing the cursor in its final position.
- Multi-value separators
- Security options, such as Enable encryption for this field.

Fonts and Colors Tab

Tab 4 of the InfoBox lets you specify fonts and colors for the field data. The tab looks like this:



Properties for: Field

Font: Helv

Size: 9

Style: ✓Plain

Alignment Tab

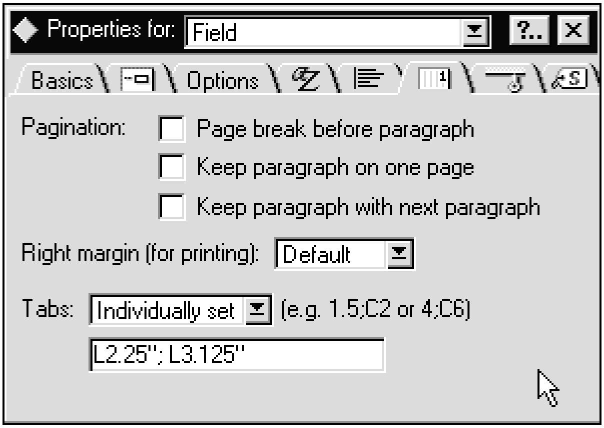
Tab 5 lets you specify the alignment of the field. For example, you should use this option if you define a field to be used as the title. If you choose to align it in the center of the form, it will stay in the center independent of the screen resolution used.

Print Option Tab

1. Go to tab 6 to set the printing defaults related to paragraphs. For example, you can select to keep a paragraph on one page, or to insert a page break before a paragraph.
2. Specify the tab settings for the fields. You must use that option if you have multiple fields on the same line and want to keep them in the same place where you have put them in the form.

   Although you can set the tabs manually, it is better to set them directly in the form using the ruler. To do so:
3. From the standard navigator, choose Design, then Forms.
4. Open the Document form.
5. Choose View - Ruler from the menu, or click on the Ruler SmartIcon.
6. Click with the left mouse button if you want a left tab stop.
7. Click with the right mouse button if you want a right tab stop.
8. Press **SHIFT** and click left if you want a decimal tab stop.
9. Press **SHIFT** and click right if you want a centered tab stop.

Here is an example of left tab stops set for the Date and TimeCreated fields using the ruler and using the InfoBox of the Date field:



Properties for: Field

Pagination: □ Page break before paragraph
□ Keep paragraph on one page
□ Keep paragraph with next paragraph

Right margin (for printing): Default

Tabs: Individually set (e.g. 1.5;C2 or 4;C6)

L2.25"; L3.125"

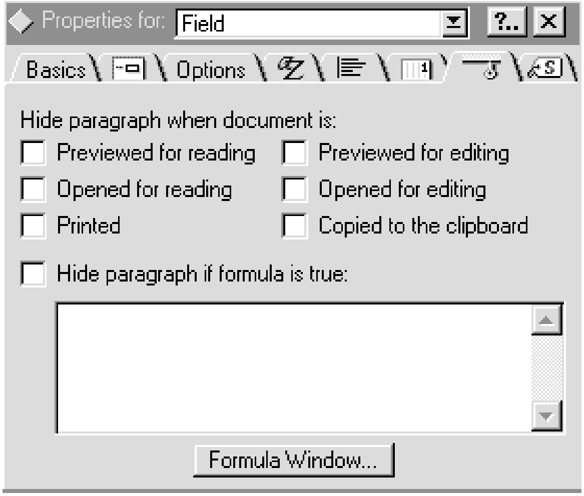
10. To remove a tab stop from the ruler, click on it.

11. If you click the tab stop on the ruler with the right mouse button, the tab stop information is displayed. From there, you can change the tab stop from one type to another.

Hiding the Field

1. Return to the InfoBox for the Categories field.

2. Go to tab 7. It looks like this:



Properties for: Field

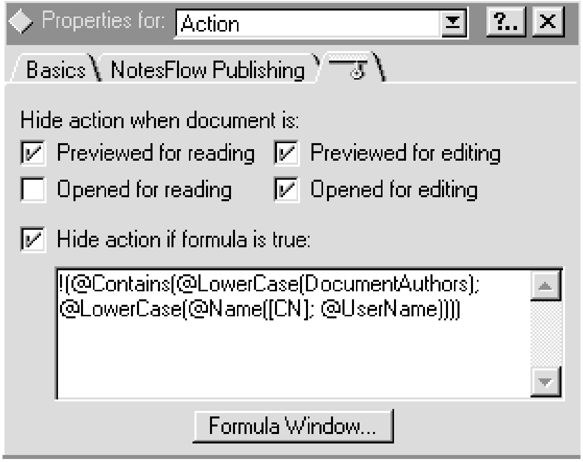
Hide paragraph when document is:

□ Previewed for reading □ Previewed for editing
□ Opened for reading □ Opened for editing
□ Printed □ Copied to the clipboard

□ Hide paragraph if formula is true:

3. Several checkboxes are available to hide the paragraph on predefined conditions. Or, you can specify other conditions using an @Function.

   The following example shows the design of the Edit Document button, which is available on the action bar when you open a document or a response to a document:



Properties for: Action

Hide action when document is:

☑ Previewed for reading ☑ Previewed for editing

□ Opened for reading ☑ Opened for editing

☑ Hide action if formula is true:

```
!(@Contains(@LowerCase(DocumentAuthors);
@LowerCase(@Name([CN]; @UserName))))
```

   The button is hidden when the document is previewed for reading, previewed for editing, and opened for editing. It is also hidden if the user is listed in the @Function as specified in the formula box. In all other cases, the button is visible.

   Note The InfoBoxes of all the design elements found in a form provide a tab that allows you to specify hide-when conditions.

4. Return to the standard navigator.

Creating Design Elements for Subforms

You have seen that you can share and reuse fields across several forms. Subforms, which are new in Release 4, extend the reuse of fields to groups of design elements.

All design elements that are added to forms can also be put into subforms. This includes:

- Static text and pictures
- Fields, whatever their type and format
- Hotspots as buttons or links
- Tables
- Action Bar

When you modify an existing subform, the changes are immediately reflected in all the forms that use the modified subform.

Note You cannot create subforms within subforms.

Looking at a Subform

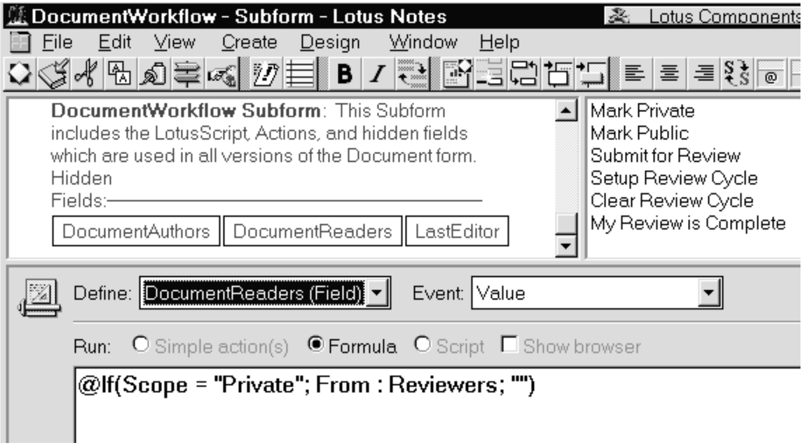
A subform is provided with the Document Library template. You can reach its design using one of the following ways:

1. From the standard navigator, choose Design, then Subforms. The list of subforms is displayed. Double-click on the DocumentWorkflow subform. The Subform Builder window is displayed.
2. Or, from the standard navigator, choose Design, then Forms. The list of forms is displayed. Double-click the Document form. Once the form is open, double-click the subform part of the form. The Subform Builder window is displayed.

   Tip You might have to scroll up to the top of the window to see the subform part.

The following figure shows you that the Subform Builder window is identical to the Form Builder window. It contains:

- The form in the design pane.
- The actions linked to the subform in the action pane. When a form and a subform are displayed, the action bars of both the form and the subform are shown.
- The field definition in the programming pane. In subforms as in forms, both @Functions and LotusScript can be used.



DocumentWorkflow Subform: This Subform includes the LotusScript, Actions, and hidden fields which are used in all versions of the Document form.

Hidden Fields:

DocumentAuthors | DocumentReaders | LastEditor

Mark Private
Mark Public
Submit for Review
Setup Review Cycle
Clear Review Cycle
My Review is Complete

Define: DocumentReaders (Field) Event: Value

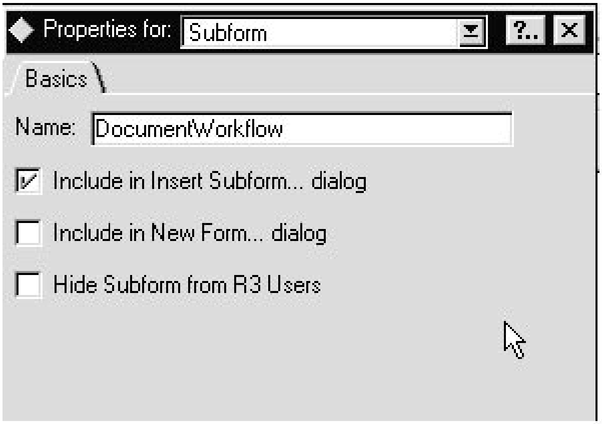
Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

```
@If(Scope = "Private"; From : Reviewers; "")
```

Looking at the Subform Properties

To display the subform properties:

1. On the subform pane, click your right mouse button.
2. Select Subform Properties. The InfoBox is shown. It looks like this:



Properties for: Subform

Basics

Name: DocumentWorkflow

☑ Include in Insert Subform... dialog

□ Include in New Form... dialog

□ Hide Subform from R3 Users

3. If required, check the Hide Subform from R3 Users check box.

   Note You must hide the subform from Release 3 users if the subform contains features that are not available in Notes Release 3, such as layout regions.

4. Close the InfoBox.
5. Close the subform.

Removing Subforms

You can remove subforms from the design of a form or from the design of a database.

Removing Subforms From the Form Design

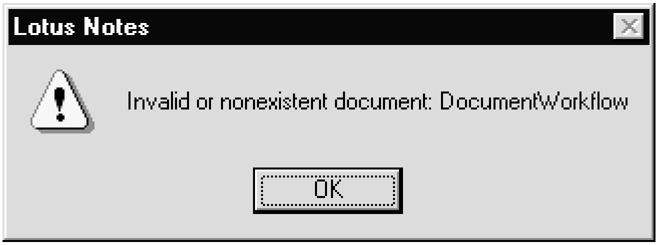
If the subform is no longer needed in a particular form:

1. Open the design of the form.
2. Click on the subform area.
3. Choose Edit - Clear on the menu bar.

Removing Subforms From the Database Design

As for all design elements, you can remove subforms from the design of the database. This can be necessary if all the fields contained in the subform are no longer needed in any of the database forms.

However, if the database contains documents using the deleted subform, this is the message users will see when they try to browse the documents.



Lotus Notes

Invalid or nonexistent document: DocumentWorkflow

Note To make sure the users can still access the documents, you must have an empty subform using the same name as the deleted subform.

Working With Layout Regions

A layout region is generally used as a nice background of a form. It consists of a 16-bit graphic that contains several kinds of design elements, such as fields (except for rich text), static text, buttons, and others.

Creating a Layout Region

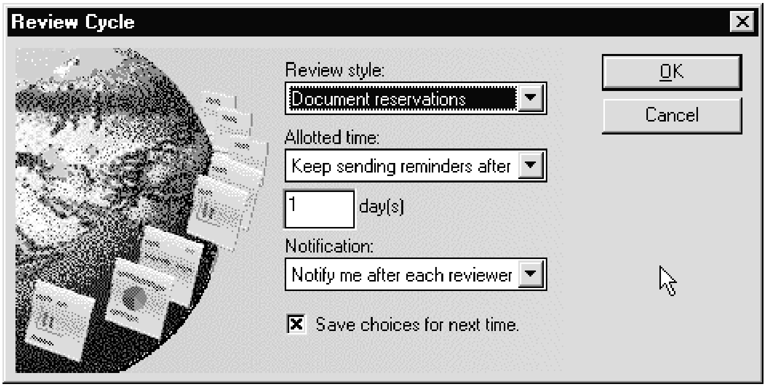
To create a layout region:

1. From the standard navigator, choose Design, then Forms.
2. Open up a form listed in the view pane, for example, the Document form.
3. Position the cursor in an empty area.
4. Choose Create - Layout Region - New Layout Region from the menu bar. An empty frame is built in the form.
5. Open up a Picture tool such as Windows PaintBrush.
6. Open a BMP file.
7. Copy it to the clipboard.
8. Switch back to Lotus Notes.
9. Choose Create - Layout Region - Graphic. The graphic is pasted from the clipboard to the layout region.
10. To add static text, choose Create - Layout Region - Text. A frame is created within the graphic. You can move the static text frame around. If you double-click on the static text frame, you display the Control box.
11. To create fields, choose Create - Fields as in any other form.
12. Return to the standard navigator.

Let’s Look at an Example

An example of a layout region exists in the Document Library template. To look at it:

1. From the standard navigator, click All Documents in the navigator pane. The list of available documents is displayed in the view pane.
2. Open the document entitled My first document.
3. Click on the Edit Document button on the action bar.
4. Click on the Setup Review Cycle button on the action bar. The Layout Region document is displayed.
5. In the Review style field, choose Document reservations.
6. In the Allotted time field, choose Keep sending reminders after.
7. In the day field, type 1.
8. In the Notification field, choose Notify me after each reviewer.
9. Check the Save choices for next time check box. The completed Review Cycle window looks like this:



Review Cycle

Review style: Document reservations

Allotted time: Keep sending reminders after

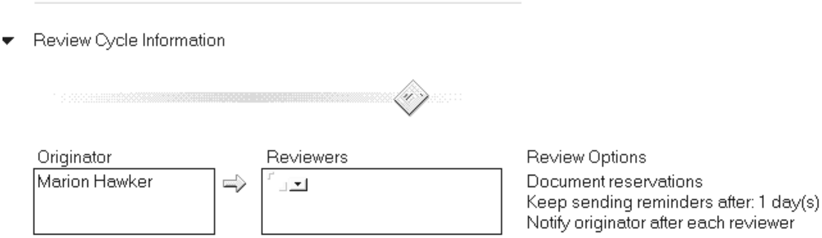
1 day(s)

Notification: Notify me after each reviewer

☒ Save choices for next time.

10. Click OK to create the review cycle. This takes you back to the document. Notice that a new element called Review Cycle Information has been added to the document. This is a collapsible section, which we will look at later on in this chapter.

    The new element looks like this:



Review Cycle Information

Originator: Marion Hawker

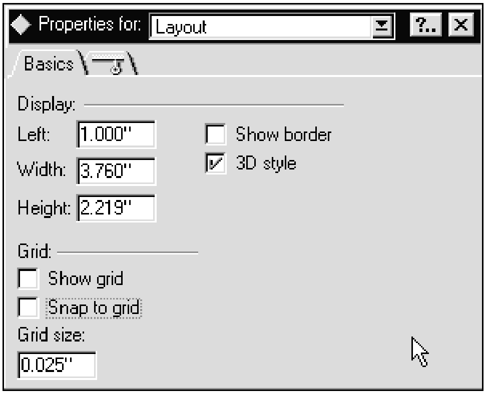
Reviewers:

Review Options
Document reservations
Keep sending reminders after: 1 day(s)
Notify originator after each reviewer

Looking at the Properties of a Layout Region

To open the layout region that was used to create the Review Options form, follow these steps:

1. The Document Library database should still be open. From the standard navigator, choose Design, then Forms. The available forms are listed in the view pane.
2. In the view pane, double-click on ReviewOptions to open up the form. It contains the layout region you were looking at earlier.
3. Click on bitmap.
4. Click the Properties SmartIcon. The following InfoBox is shown:



Properties for: Layout

Basics

Display:

Left: 1.000"

Width: 3.760"

Height: 2.219"

□ Show border

☑ 3D style

Grid:

□ Show grid

□ Snap to grid

Grid size: 0.025"

The InfoBox has two tabs:

5. On the Basics tab, you can adjust the dimension of the layout region. You can also display the grid to position the fields and static text within that region.
6. On the Hide tab, you can select to hide the design element. Several options are available. You can also define an @Formula to hide the design element.
7. Close the InfoBox.

Working With Collapsible Sections

If the form design includes a long set of fields or fields that contain large amounts of data, it can be annoying for users to scroll up and down to find the information they are looking for.

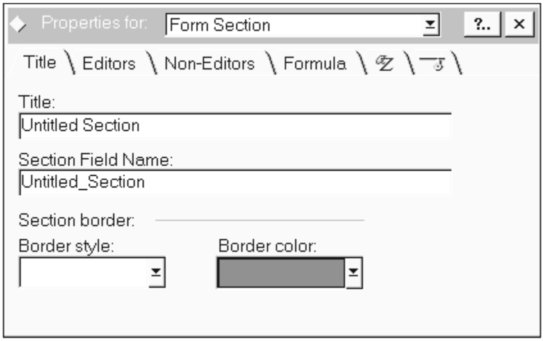
In Notes Release 4, you can create collapsible sections to solve that problem.

Creating a Collapsible Section

To create a collapsible section within a form, follow these steps:

1. Open the design of a form.
2. Choose Create - Section - Standard if you want the section to be seen by all the users that have access to the document.
3. Or, choose Create - Section - Controlled Access if you want to restrict access to the section to certain users defined in a formula.

   The InfoBox for Form Section looks like this:



Properties for: Form Section

Title: Untitled Section

Section Field Name: Untitled_Section

4. Use the Editors tab to choose expand and collapse rules for users who have editor access.
5. Use the Non-Editors tab to choose expand and collapse rules for users who do not have editor access.
6. The Formula tab enables you to specify the required formula for controlled access to the section.
7. Close the InfoBox.

Looking at a Collapsible Section

When you were working on the layout region design, you noticed that specifying the review cycle information actually created a new section in the document that was being edited.

If you look at what was performed in the Setup Review Cycle action, you will see that the macro does the following (see the DocumentWorkflow subform to get the design of this action):

- Receives the user input using the @DialogBox function:

```
@If( @DialogBox("ReviewOptions";
               [AutoVertFit]:[AutoHorzFit];
                "Review Cycle") = @False;
     @Return(@False);
     "");
```

- Stores data in fields such as:

  ```
  FIELD Status:= @Subset(@Subset(StatusList;2);-1);
  ```

- Expands the Section:

  ```
  @Command([SectionExpandAll]);
  ```

- Positions the cursor in a specific field:

  ```
  @PostedCommand([EditGotoField]; "Reviewers");
  ```

- Refreshes all fields:

  ```
  @PostedCommand([ViewRefreshFields])
  ```

  The purpose of this field refresh is twofold:

  - To make sure that the user sees the data modified by the macro
  - To show the collapsible section that was previously hidden because of the value of the Status field. This is the formula specified on the Hide page of the Properties box for the section:

    ```
    @Member(Status; tmpStatusList) = 1 : 4 | @IsResponseDoc
    ```

To see the design of the section Review Cycle Information, perform the following steps:

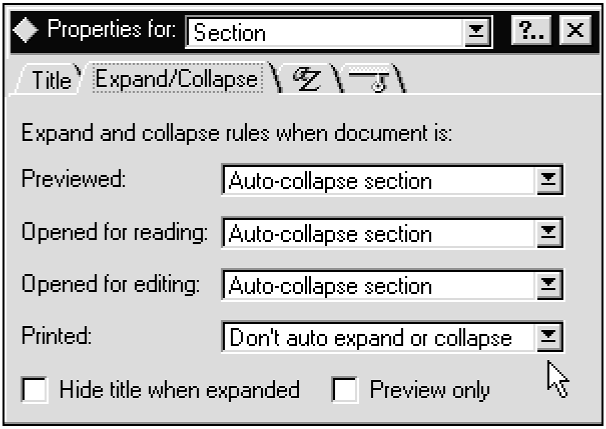
1. From the standard navigator, choose Design, then Forms. The available forms are displayed in the view pane.
2. Double-click on Document to open up the form containing the collapsible section.

   You can see an arrow next to the Review Cycle text. This is a twistie: if you click on it, the section expands and shows all the section information. If you click again, it collapses.

Looking at the Properties of a Standard Collapsible Section

1. Open the form for which you created a standard collapsible section earlier. Or, create a standard collapsible section now before proceeding. For example, you can define a section for the Document form.
2. To display the properties of the collapsible section, put the cursor in the section area, and click the Properties SmartIcon. The Properties for Section InfoBox is displayed. It consists of four tabs.

3. The Expand / Collapse tab enables you to define expand and collapse rules when the document is previewed, opened for reading, opened for editing, or printed.



Properties for: Section

Expand and collapse rules when document is:

Previewed: Auto-collapse section

Opened for reading: Auto-collapse section

Opened for editing: Auto-collapse section

Printed: Don't auto expand or collapse

□ Hide title when expanded □ Preview only

Tip If a document based on this form can be read by many users but created by only few users, you could have the section automatically expanded if the document is opened in edit mode, and automatically collapsed if it is opened in browse mode.

Working With Tables

Tables are added to forms to format data as columns and rows. Within the table columns and rows, you can create design elements such as fields and buttons.

For example, suppose you want to create a series of buttons next to each other. You do not want the buttons to be split on several lines when the window is resized.

To accomplish this, follow these steps:

1. In your form, click where you want to create the table.
2. Choose Create - Table.
3. In the Rows field, type 1.
4. In the Columns field, type 1.
5. Click in the table area.
6. Click the Properties SmartIcon to display the InfoBox.

7. On the Cell Borders tab, click Select All to None.
8. On the Layout tab, deselect Fit to Window.
9. Start creating the buttons by choosing Create - Hotspot - Button, and place all the buttons inside the table.

Working With Buttons

The action bar is a new and convenient way for you to position buttons that the user of the database will most often use. The action bar is available in both forms and views.

This chapter shows you how to work with buttons in forms. The following chapter covers buttons in views.

In a form, you can have buttons to:

- Switch from browse to edit mode.
- Close the document with or without saving it.
- Issue actions that are accessible even if the user scrolls down the document.

The only restriction is the width of the action bar. It cannot be scrolled horizontally and it does not expand across several lines. Therefore, it is recommended that you:

- Put only the most common actions on the action bar.
- Decide whether or not to put an icon on the action bar. No icons means additional space for more action buttons.
- Keep the action title short though meaningful.
- Develop for a VGA display if your users have different screen resolutions, such as VGA 640 pixels and SVGA 800 and 1024 pixels.

Creating a Button on the Action Bar

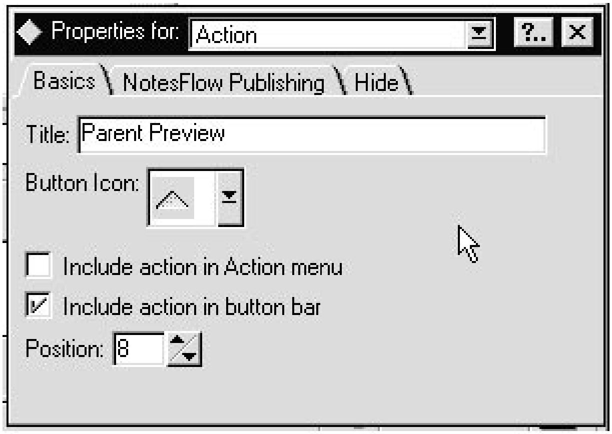
We have seen earlier in this chapter that you can relate a response document to its parent using a DocLink, which requires very little code, namely, one @Function.

But there is one small inconvenience: when users click on the DocLink, the parent document is displayed in full-screen hiding the response document they started from.

It would be easier to look for information if both the response document and its parent were visible at the same time.

You can achieve this by using the parent preview feature.

1. From the standard navigator, choose Design, then Forms. The list of forms is displayed in the view pane.
2. Double-click on the Response from. The form is displayed.
3. Choose Create - Action from the menu bar. An InfoBox is displayed. It looks like this:



Properties for: Action

Title: Parent Preview

Button Icon:

☐ Include action in Action menu

☑ Include action in button bar

Position: 8

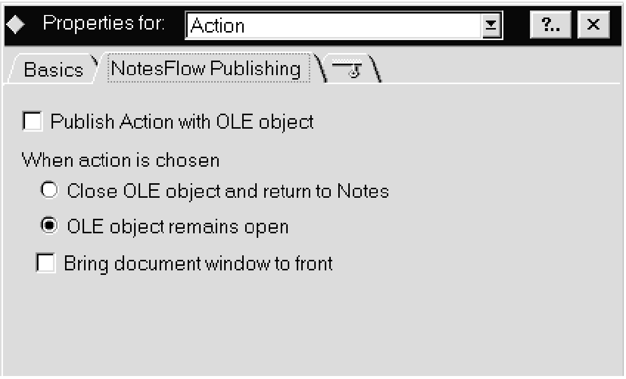
4. In the Title field, type Parent Preview, for example.
5. From the Button Icon list, choose an icon for the button.

   Note You cannot add your own icon.
6. Deselect Include action in Action menu.
7. If desired, change the position the button will have on the action bar, or leave the value provided by Notes.
8. Close the InfoBox.
9. In the programming pane, type the following @Command:

   ```
   @Command([ShowHideParentPreview])
   ```

   This @Command toggles the parent document preview pane.
10. Choose File - Save, then File - Close.

Note The NotesFlow Publishing tab of the InfoBox lets you publish the action with an OLE object. The tab looks like this:



□ Publish Action with OLE object

When action is chosen

○ Close OLE object and return to Notes

● OLE object remains open

□ Bring document window to front

You are back on the view pane.

11. Select All Documents to display the list of documents.
12. Double-click on the Response 1 document to open it.
13. At the top of the document, the Parent Preview button is displayed on the action bar.
14. Click on it. The window is divided into two parts, with the parent document shown below the response document.

Note So far, you have looked at a DocLink connecting a parent with its child document. You can also create a DocLink between documents that are not related in that way. This can be very useful if you need to look up information in documents within the same database or even across databases.

Creating Hotspots

When designing forms, you can create the following types of Hotspots:

- Links
- Text Popup
- Buttons

Creating Links

You can create links:

- When designing the form
- When creating or saving a document in the database

Links enable navigation to:

- Notes documents

  You did this earlier when creating a DocLink between a parent and a child document. See the section “Creating a New Field.”

  You can also create the link when you design a form. This could be useful if you want to enable your users to quickly access a document in a reference database, for example.

- Notes databases

  While designing a form, you can define a link to an existing database for quick access by the users.

  This icon represents a link to a database:

- Internet Web pages

  You can create a link to a Web page by making a hotspot to its URL and using the @URLOpen command.

Defining Text Pop-ups

When you design a form, you can define text pop-ups to complement the available online help information.

You are going to create a text pop-up for the DocLink field you created in the Response document.

1. From the standard navigator, choose Design, then Forms.
2. On the view pane, double-click on Response.
3. Mark the text Document link: using the cursor.
4. Choose Create - Hotspot - Text Pop-up from the menu bar. The Hotspot Box is displayed.
5. The Hotspot Box is drawn. On the first tab, type some text in the Pop-up Text field. This is the text that users will see when they keep mouse button 1 pressed over the Document Link text.

Creating Buttons in Forms

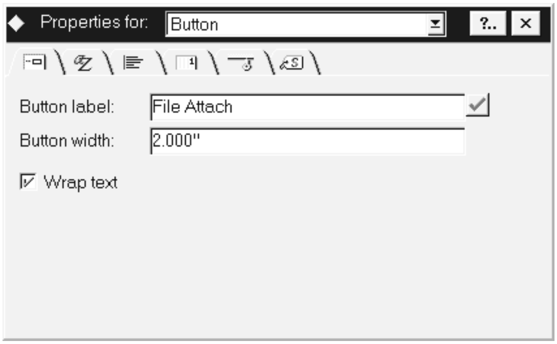
You have seen that buttons can be put on action bars. They can also be put into a form. A button contained in a form performs a menu command or an action whenever the user clicks on it.

Actions can be:

- Calculation of data
- Lookups of data in current or other databases using the @DbLookup, @DbColumn, @PickList formulas
- Written using @Functions or LotusScript

Since the Document Library database is used to store documents, you are going to create a button next to the Body field in the Document form. This button will replace the File - Attach option.

1. On the navigator pane, choose Design, then Forms.
2. Double-click on the Document form.
3. Position the cursor above the Body field.
4. Choose Create - Hotspot - Button from the menu bar. The Properties for Button InfoBox is displayed. It looks like this:



Properties for: Button

Button label: File Attach

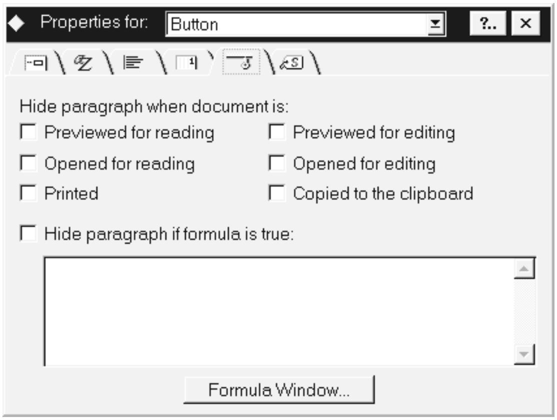
Button width: 2.000"

☑ Wrap text

5. On the first tab, type a button label, for example, File Attach. You can also adjust the button width.
6. On the second tab, you can select the font and the color of the button text.
7. The third tab enables you to specify alignment options.
8. Use the fourth tab to define pagination options, and to position the button using the Tabs field.

   Tip It is actually easier to specify tabs using the ruler.

9. The fifth tab lets you define Hide paragraph options. It looks like this:



Properties for: Button

Hide paragraph when document is:

☐ Previewed for reading ☐ Previewed for editing

☐ Opened for reading ☐ Opened for editing

☐ Printed ☐ Copied to the clipboard

☐ Hide paragraph if formula is true:

10. Check Previewed for reading and Opened for reading, since attachments can only be done when the document is in edit mode.

11. In the programming pane, type the following macro:

    ```
    @Do( @PostedCommand([EditDown]; "1");
         @PostedCommand([EditInsertFileAttachment]))
    ```

    The macro tabs the cursor down one field and performs a menu command, which is File - Attach.

12. Close the form design and save it.

To test the macro button:

13. On the view pane, select All Documents.

14. Click on the New Document button on the action bar.

15. Once the document is open, you can click on the Attach File button you have just created. The Create Attachment(s) InfoBox is displayed that allows you to copy files into the Notes document.

Tip For more information on forms and fields, refer to the appropriate chapters in the *Application Developer's Guide* provided with Lotus Notes.

Chapter 5
Viewing the Database

When you are done with this chapter, you should know:

- About the structure of views, folders and navigators:
  - How to create and modify them.
  - How to edit their properties.
  - How to create an action bar.
- How to move documents from views and folders.
- How to control the information displayed in a view:
  - Document Selection.
  - Field Selection.
- How to control who has access to views.

What Is a View?

Views list the documents stored in a Notes database. Views are tables of contents of the database. Each row listed in the view represents data taken from a single document. Each column represents a field or a combination of fields taken from that document.

All Notes databases have at least one view. Most of them have more than one view. A view can display all the documents, or it can display a subset of the documents. Documents can be viewed by categories, such as creation date and author. Views can present documents sorted on different fields, for example, sorted by topic or document reference number.

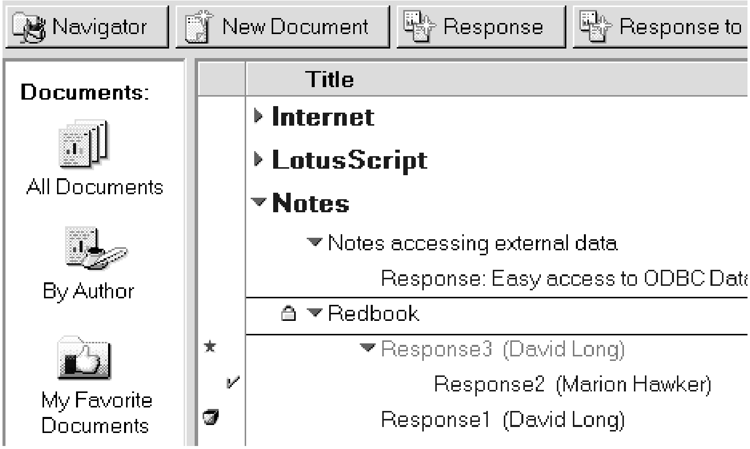
Changing an Existing View

The Document Library template contains a lot of the new Notes Release 4 features such as folders, a navigator, an action bar, and new features related to views. You are going to look at one of its views, and change parts of it.

The Document Library template should be on your Notes workspace. You added it to your workspace in the Designing Application Forms chapter.

Looking at a View

1. Open the Document Library database.
2. Click the Navigator button. This is what you can see:



Title
Internet
LotusScript
Notes
Notes accessing external data
Response: Easy access to ODBC Data
Redbook
Response3 (David Long)
Response2 (Marion Hawker)
Response1 (David Long)

The action bar is displayed at the top of the window. It contains buttons that perform either simple actions or @Formulas. The Navigator button toggles the display between the Navigator view and the default standard navigator, which is also known as the outline hierarchy.

The navigator pane displayed on the left-hand side is a graphical representation of views. In our example, you can see the All Documents view. In fact, each icon of the navigator is linked to a specific view that exists in the database.

3. Click on any of the icons displayed in the navigator pane. Notice how the view pane on the right-hand side changes.

The view pane lists the documents in a table format. The view pane consists of two parts: the selection column and the documents themselves.

The selection column shows that the document entitled Response 3 has not yet been read. Response 2 has been selected, either by a user or as the result of a search action. Response 1 has been marked for deletion.

The documents are categorized and sorted, depending on the contents of the Categories field contained in the Document form.

The Internet and LotusScript categories are collapsed, that is, you do not see the documents listed. The Notes category is expanded. You see its documents and responses to these documents.

Next to the Redbook document, an icon representing a lock is displayed. It indicates that this document is a private document. Notes can represent data using either text or a predefined set of icons.

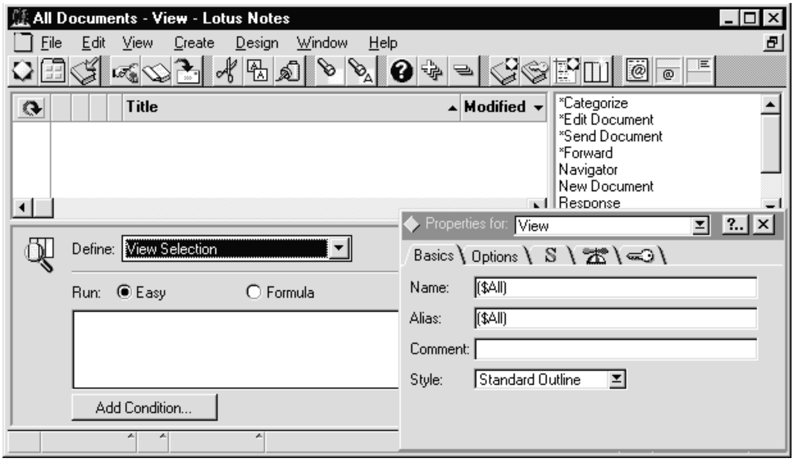
Looking at the View Design

The Document Library database should still be open.

1. Click on the Navigator button to return to the standard navigator.
2. Choose Design, then Views. The available views are listed in the view pane.
3. Double-click on the ($All) view to open its design.

   Note ($All) is a reserved name for Notes to create the default view All Documents with its associated icon.

   The View Builder window is displayed.
4. Click on the Properties SmartIcon to display the View properties. The View Builder window and the InfoBox look like this:



Properties for: View

Define: View Selection

Run: ● Easy ○ Formula

Name: ($All)

Alias: ($All)

Comment:

Style: Standard Outline

You can see the following:

- The design pane on the top left-hand side, with the view columns listed under the SmartIcons.
- The programming pane at the bottom of the window.
- The action pane on the top right-hand side.

Tip You might need to scroll the right side of the window to the left to see the action pane.

Looking at the View Columns

Below the row of SmartIcons, you can see a blue arrow icon and the columns of the view.

This refresh icon is used to look at the result of any design changes you might have performed. It allows you to test your changes before saving the new version of the view.

The first four data columns do not have any title. The remaining columns show a title. Those are the column titles users will see when they open the view.

Column editing and column properties are covered later in this chapter.

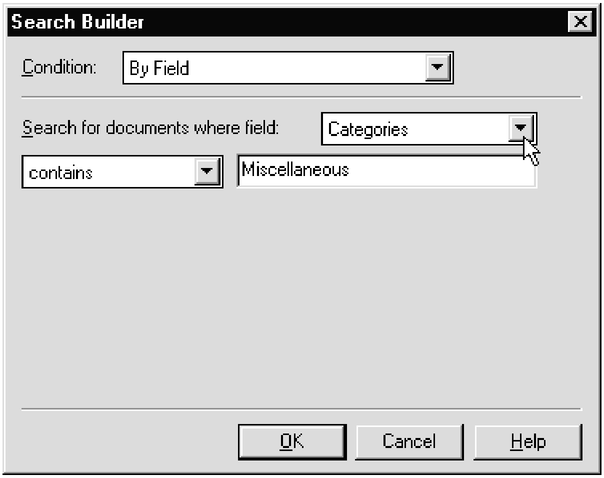
Working with the Programming Pane

In the programming pane at the bottom of the window, the View Selection that defines the view documents is empty. This means that all documents in this database are listed in the view.

To define a more restrictive view selection:

1. Click the Formula option button.
2. Click the Fields and @Functions push button. It gives you access to:
   - The fields of all the database forms.
   - All the @Functions with their help documents.
3. Or, click the Easy option button.
4. Click on the Add Condition...push button. The Search Builder window is displayed. It allows you to specify search criteria.
5. For example, in the Search for documents where field: section, select Categories from the drop-down list.
6. In the contains section, type Miscellaneous.

   The following figure shows you what it looks like:

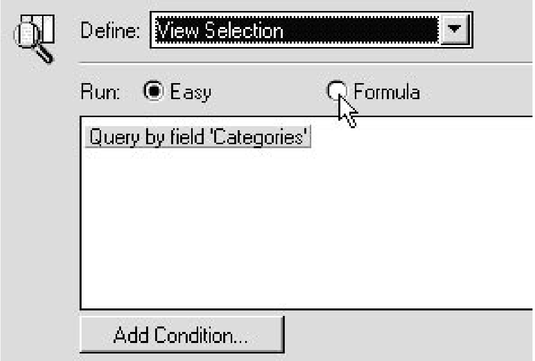


Condition: By Field

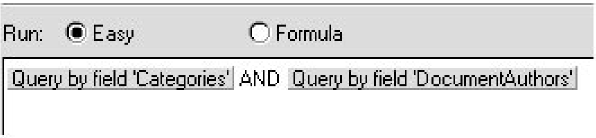
Search for documents where field: Categories

contains Miscellaneous

7. Click on OK. Notice how the programming pane has changed to reflect the new search criteria.



8. Click again on the Formula option button. The formula and the appropriate @Functions are displayed in the programming pane.
9. Use the Easy option button and Add Condition button to specify as many search criteria as you want, for example:


Run:
● Easy
○ Formula
Query by field 'Categories' AND Query by field 'DocumentAuthors'

10. When you are done, close the Search Builder window.

Working with the View Properties

To display the View InfoBox:

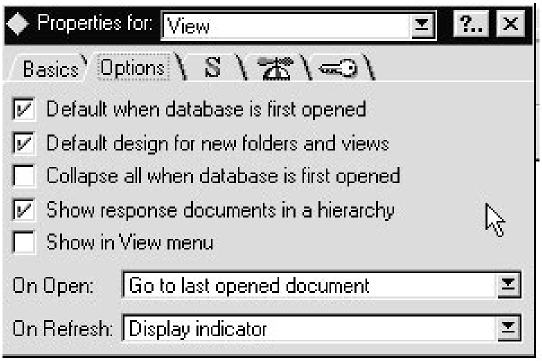
1. Click on the Properties SmartIcon to display the InfoBox. It contains five tabs.

   On the Basics tab, the Name of the view is set to a reserved word, ($All). The user will see the name as All Documents in the View menu item.

   Except for the reserved ($All), enclosing a view between parentheses means that the view is hidden and is used solely for programming purposes: The user does not see it in the list of views.
2. Specify an alias. You should always use the alias. This is the name you will use in your code. If the user decides to change the name of the view, your code will remain unchanged.
3. Specify a comment. The comment entry field is optional, but useful for maintenance purposes.

4. Choose a style. You can display the documents in a view as a calendar instead of as a table. For example, a calendar view can display a date, a meeting or appointment time, a duration, and optional text describing the entry. To display a view as a calendar the first column must be a Time / Date field.

5. Go to the Options tab. It looks like this:



☑ Default when database is first opened
☑ Default design for new folders and views
☐ Collapse all when database is first opened
☑ Show response documents in a hierarchy
☐ Show in View menu

On Open: Go to last opened document

On Refresh: Display indicator

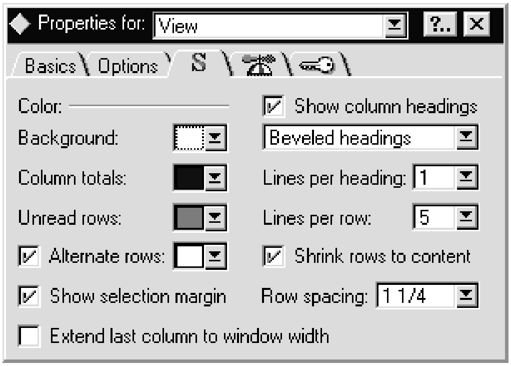
If you check Default design for new folders and views, it means that the view will be used as the template when the user creates folders or adds new views to the database.

The check next to the Show Response documents in a hierarchy will show all child documents indented under their parent document.

If the Show in View menu checkbox is not selected, it means that the view is hidden to the users and only used by the application for lookup purposes.

6. Go to the Style tab.

Several new features have been introduced in Notes Release 4 that you find in this InfoBox tab.



Color:
Background: □
Column totals: ■
Unread rows: ▨
☑ Alternate rows: □
☑ Show selection margin
☐ Extend last column to window width

☑ Show column headings
Beveled headings
Lines per heading: 1
Lines per row: 5
☑ Shrink rows to content
Row spacing: 1 1/4

7. In addition to the color that can be customized for the view background, columns totals and unread rows, you can now define a color to be used for alternate rows. And there are a lot more colors to choose from.

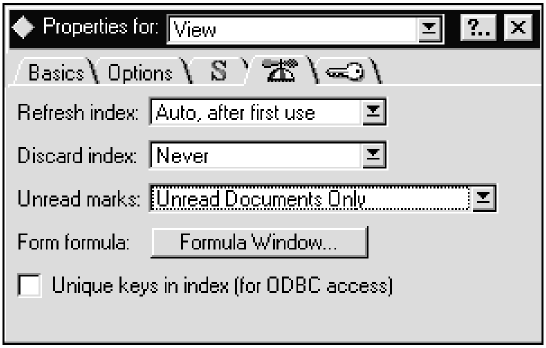
8. If desired, you can remove the selection margin.

9. You can now have up to 5 lines for the column headings.

10. As far as rows are concerned, you can also have up to 9 lines for each document, to store a large description for instance. In that case, make sure that the checkbox Shrink rows to content is selected: It eliminates all the blank lines for documents that do not require the extra space.

11. Go to the Advanced tab.

    The advanced tab provides information about the index used to build the view, when it should be refreshed (here automatically) and when it should be discarded.



Refresh index: Auto, after first use

Discard index: Never

Unread marks: Unread Documents Only

Form formula: Formula Window...

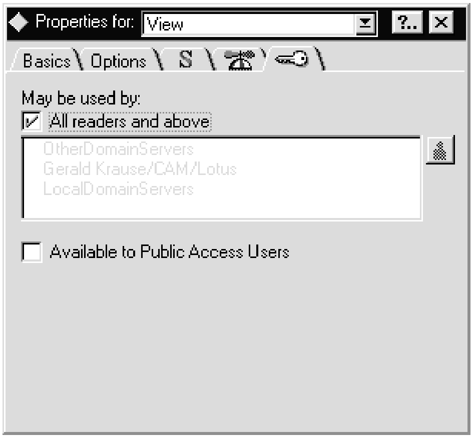
□ Unique keys in index (for ODBC access)

    Users will see highlighting next to documents that were added or modified since they last used the view. The ODBC checkbox is not selected as there is no unique key built in the view.

12. Go to the Security tab.

    In this example, the view can be used by all users that have access to the database. If you want to restrict its use to only some users or groups of users, deselect the checkbox and add the users or groups that will be granted access.

Also, you may select the Available to Public Access Users option to enable non-Notes users to access the view.



May be used by:

☑ All readers and above

OtherDomainServers
Gerald Krause/CAM/Lotus
LocalDomainServers

□ Available to Public Access Users

13. Close the InfoBox.

Creating Views

Now that you have seen how to browse and change some of the settings of a view, here is how to create a new view. There are two ways:

Copying an Existing View

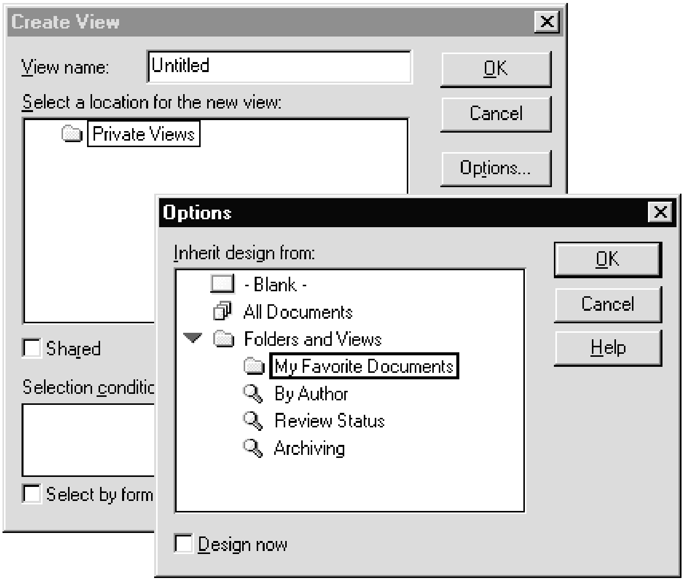
To copy an existing view:

1. Open the database where the view you want to copy is located. It can be in the same database or in another database.
2. On the standard navigator, click on Design, then Views.
3. In the view pane, select the view you want to copy.
4. Choose Edit - Copy to copy the view to the clipboard.
5. Open the database where you want the new view to be created.
6. On the standard navigator, click on Design, then Views.
7. Click on the view pane, then choose Edit - Paste to copy the contents of the clipboard. The new view is created.

Creating a New View

To create a new view:

1. Open the database to display the standard navigator.
2. Choose Create - View. The Create View dialog box is displayed. All the settings for the new view are defaulted to those of the default view as defined in the Options tab of the view properties.



Create View

View name: Untitled

Select a location for the new view: Private Views

□ Shared

Selection conditic

□ Select by form

Options

Inherit design from:

- Blank -
All Documents
Folders and Views
My Favorite Documents
By Author
Review Status
Archiving

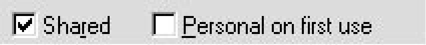
□ Design now

3. Change the view name. Including backslashes in the name will cascade the views in the View Menu. For example: Marketing\Lotus.
4. If you want to use another view as the default, click on the Options push button. The Options dialog box is displayed. You can now select a different view design as the default for your new view.

   Note The view created here is a private view. If you want to create a view available to all users, do not click the Options push button but instead select the Shared checkbox. The shared views are then listed.
5. If you select the Design now checkbox, the design windows of the new view are displayed so that you can start your customization immediately. Otherwise click on the OK button.

6. Click on the item Views to store the view you are creating at the top level of the list or click on the name of another view under which the new view should appear.

7. If you want your users to have a common private view, select the checkbox Personal on first use.



☑ Shared ☐ Personal on first use

The view is shared, which means it is available to all users, but it turns private as soon as the users access the database.

This is a very effective way to develop and distribute private views to multiple users.

Editing the View Columns

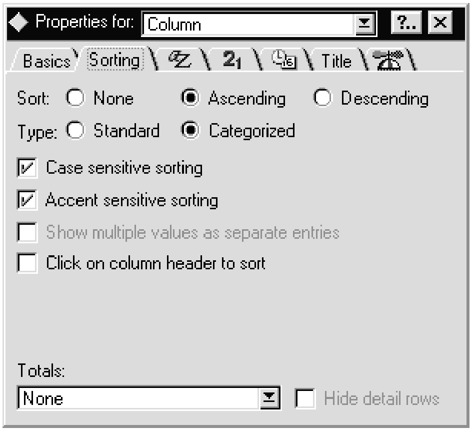
You are going to edit a few of the columns of the view All Documents to see some of the column properties.

1. On the standard navigator, click on Design, then Views.
2. Double-click on ($All). The view edit window is displayed.

Categorization

Double-click on the first untitled column to display both the InfoBox to show the column properties and its values in the programming pane.

The Sorting tab of the InfoBox is displayed in the figure below.



Properties for: Column

Sort: ○ None ● Ascending ○ Descending

Type: ○ Standard ● Categorized

☑ Case sensitive sorting

☑ Accent sensitive sorting

☐ Show multiple values as separate entries

☐ Click on column header to sort

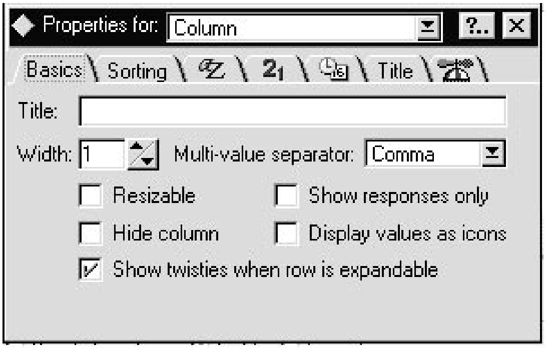
Totals: None ☐ Hide detail rows

The programming pane shows that the column is equivalent to the value contained in the Categories field.

The Type option button shows that the column is categorized, which means:

- The documents are sorted on the field Categories.
- The value of the field Categories is not displayed on every row but as a header to the documents in that category.

This is why, on the Basics tab shown in the following figure, the width of that column can be set to 1 character. Also, the twistie checkbox is selected. When a user displays the view, an arrow will be shown next to those categories that can be expanded.



Properties for: Column

Title:

Width: 1 Multi-value separator: Comma

□ Resizable □ Show responses only

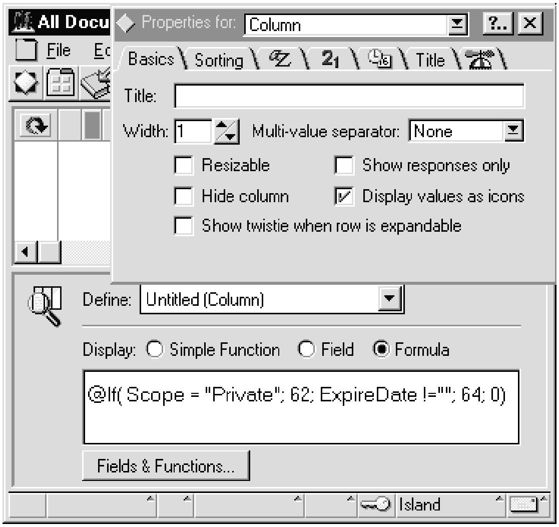
□ Hide column □ Display values as icons

☑ Show twisties when row is expandable

Select the Hide column checkbox if it is only used for programming purposes (for example, sorting or lookups).

How About Some Icons?

Double-click on the second untitled column to display the InfoBox for that column. It looks like this:



Properties for: Column

Title:

Width: 1 Multi-value separator: None

□ Resizable □ Show responses only

□ Hide column ☑ Display values as icons

□ Show twistie when row is expandable

Define: Untitled (Column)

Display: ○ Simple Function ○ Field ● Formula

```
@If( Scope = "Private"; 62; ExpireDate !=""; 64; 0)
```

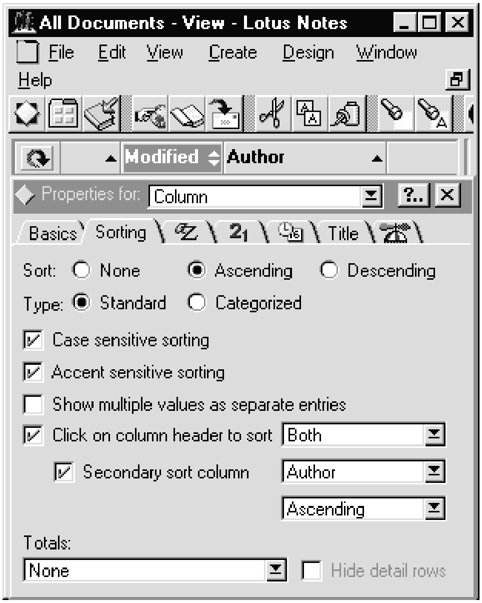
The Display values as icons checkbox has been checked. This means that the column data is displayed as icons. This is why the width of the column has a value of 1.

In the programming pane, @Functions are used to set the value of the column. If the value of the field Scope is set to Private**,** then the icon referenced as number 62 will be displayed in the view next to the document. If the field ExpireDate is not empty, the icon with the number 64 will be displayed. Otherwise, no icons will be displayed (value equal to 0).

Tip All the number references of the available 170 icons are listed in the document *Details: Displaying an icon in a column* of the Help database. Notice that you cannot add any icons to the predefined set.

Multiple Sorting

1. Double-click on the Modified column. You are going to modify the sort criteria of the column to the settings shown in the following picture:


All Documents - View - Lotus Notes
File Edit View Create Design Window
Help
Modified
Author
Properties for: Column
Basics
Sorting
Title

Sort: ○ None ● Ascending ○ Descending

Type: ● Standard ○ Categorized

☑ Case sensitive sorting

☑ Accent sensitive sorting

☐ Show multiple values as separate entries

☑ Click on column header to sort: Both

☑ Secondary sort column: Author, Ascending

Totals: None ☐ Hide detail rows

Follow these steps:

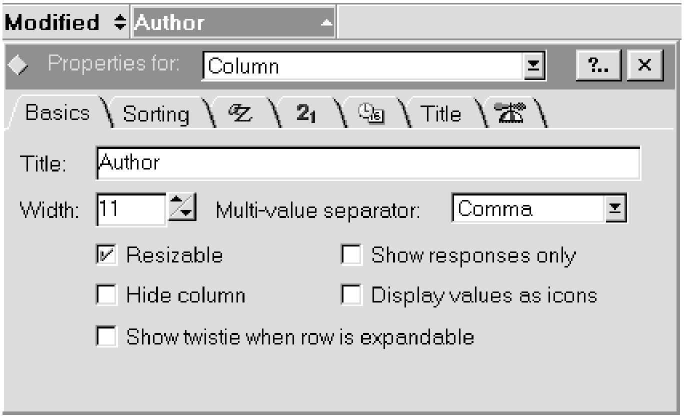
2. Drop down the list of options next to Click on column header to sort.
3. Select the Both option.
4. Check the Secondary sort column check box.

5. Select the Author option as the second-level sort criteria.

   The Modified column now has two arrows in the column heading. When a user uses this view and clicks on the upper arrow of the Modified column, the column will be sorted in ascending order. When the user clicks on the lower arrow, the column will be sorted in descending order.

Resizing the Column Width

Double-click on the Author column to display the InfoBox. The Basics tab is displayed:



Title: Author

Width: 11

Multi-value separator: Comma

☑ Resizable

□ Show responses only

□ Hide column

□ Display values as icons

□ Show twistie when row is expandable

The Resizable checkbox is selected. This means that users will be able to resize the column by dragging its border if they want to see more than the 11-character width defined by the developer.

Switching to Another View

The InfoBox with the properties for the Author column should still be displayed.

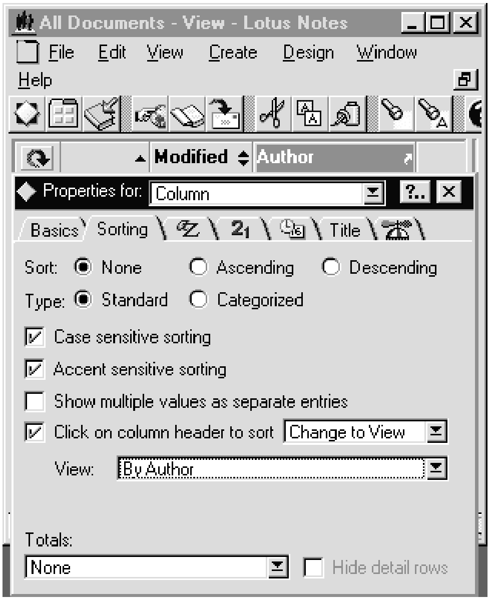
1. Click on the Sorting Tab.

   You are going to introduce a new feature of Release 4 into the view. A user looking at the All Documents view might want to take a look at the people who have created the documents. To do so, the user would typically click on the By Author view or the navigator.

   However, it is now possible to provide a different solution:

2. Drop down the list of options next to Click on column header to sort.
3. Choose the Change to View option.

4. In the view combobox that is now visible, select the By Author view from the list of the views that are available in the database. It looks like this:


All Documents - View - Lotus Notes
File Edit View Create Design Window
Help
Modified
Author
Basics
Sorting
Title

Properties for: Column

Sort: ● None ○ Ascending ○ Descending

Type: ● Standard ○ Categorized

☑ Case sensitive sorting

☑ Accent sensitive sorting

□ Show multiple values as separate entries

☑ Click on column header to sort: Change to View

View: By Author

Totals: None

□ Hide detail rows

Notice the curved arrow that now appears in the heading of the Author column. This enables users to perform a single click on the column heading to close the view they are currently working on, and switch to a new view.

Saving and Testing the View

If the database already contains documents, you do not need to save the new version of the view. Instead, you can use the Refresh icon. This lets you see the effect of the changes on the document display.

Otherwise, once you are satisfied with your modifications, press the ESC key and save the view. Now you can try all the nice features it contains:

- Multiple lines.
- Twisties.
- Multiple sort directions.
- Switch to view facility.
- Put some of the icons in columns.

Creating and Moving Columns

Copying a Column from Another View

If a column similar to the one you need already exists in another view of the same database or in a different database, you can just copy it using the clipboard:

1. Open the design of the view where the column you want to copy is located.
2. Click on the heading of the column you want to copy.
3. Choose Edit - Copy from the menu bar to store the column definition on the clipboard.
4. Open the design of the view where you want the new column to be created.
5. Click the column to the right of where you want the column to be positioned.
6. Choose Edit - Paste from the menu bar. The column is pasted left of the highlighted column.
7. Double-click on the column heading to open the InfoBox.
8. Modify the column properties as required and save the column.

Creating a New Column

You can also create a new column. To do so:

1. Open the design of the view you want to modify.
2. If you want to create the new column as the last column of the view, choose Create - Append New Column.

   If you want to create the column at a specific place in the view:
3. Click the column to the right of where you want the column to be positioned.
4. Choose Create - Insert New Column from the menu bar.
5. Double-click on the column heading to open the InfoBox.
6. Modify the column properties and save the column.

Moving a Column within the View

To move columns within the same view:

1. Open the design of the view.
2. Click on the heading of the column you want to move.
3. Choose Edit - Cut from the menu bar to store the column definition on the clipboard.

4. Click the column to the right of where you want the column to be positioned.
5. Choose Edit - Paste. The column is pasted into its new location.
6. You can then save the view or folder.

Using Folders

What Is a Folder?

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

You can keep a folder private, or share it with other users of a database. No one else can read or delete your private folders. To create private folders in a database, you must have at least Reader access to the database. To create shared folders in a database, you must have at least Designer access.

When you create a private folder, Notes stores it in one of two places:

- If the manager of the database has allowed it, your folder is stored in the database, letting you use the folder at different workstations.

  Note To see whether a database allows storage of folders, select the database, choose File - Database - Access Control, and see whether Create personal folders / views is turned on.

- If the manager has not allowed storage of folders in the database, Notes stores your folder in your desktop file.

  Note If a folder is stored in your desktop file, you cannot use full-text search in the folder.

Creating and Editing Folders and Folder Columns

Creating or editing a folder is identical to creating or editing a view. The steps are the same. See the appropriate section in this chapter for more details.

Using Folders

You can put documents into folders by dragging the documents, or by using the menu. You may want to use the menu if it is not convenient to drag, or if you want to manage a large number of documents at once.

To drag documents into a folder:

1. Select the document or documents you want to store.
2. Drag the documents to that folder's icon in the navigation pane and let go when the folder is highlighted and the cursor appears as a plus symbol (+).
3. Repeat steps 1 and 2 to drag documents to another folder.

To put documents into folders using the menu:

1. Select the document or documents you want to store.
2. Choose Actions - Move to Folder.
3. Do one of the following:

   Click a folder name in the Choose a folder list to use an existing folder. Or, to create a new folder, see Details.
4. Do one of the following:

   Click Add to put the document into a folder without removing it from other folders. Or, click Move to put the document into a folder and remove it from other folders.
5. Repeat steps 4 through 7 to put documents into more folders.

Here Again the Action Bar

As well as in forms, you can create an Action Bar in views and folders. In general, the actions should either:

- Affect several documents or all documents displayed in the view. This could be storing in a Manager folder all documents created by your manager, for example.
- Represent the actions the user will most often perform.

As in forms, you must make sure that the actions you create will fit in the Action Bar and watch out for the screen resolution available to your users.

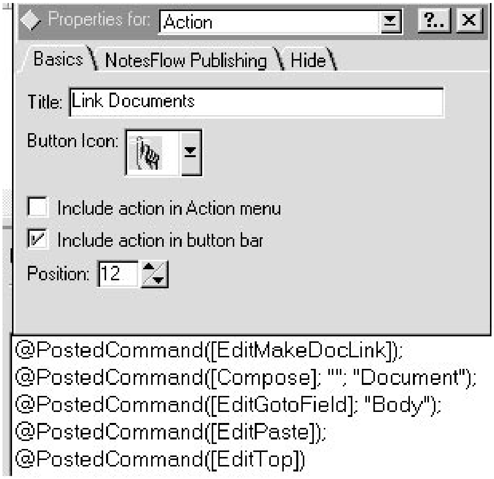
Creating a Button on the Action Bar

You are going to create an action that performs a DocLink between two documents. The documents do not have a child-parent relationship.

To create a button in the Action Bar:

1. Open the design of the ($All) view.
2. Choose Create - Action. The InfoBox for the Action properties is opened and you now have access to the programming pane.

3. Fill in the Basics tab and programming pane as shown here:



Title: Link Documents

Button Icon:

□ Include action in Action menu

☑ Include action in button bar

Position: 12

```
@PostedCommand([EditMakeDocLink]);
@PostedCommand([Compose]; ""; "Document");
@PostedCommand([EditGotoField]; "Body");
@PostedCommand([EditPaste]);
@PostedCommand([EditTop])
```

Here are some explanations concerning the formula:

The document selected from the view is copied to the clipboard:

@PostedCommand([EditMakeDocLink]);

The form Document is created:

@PostedCommand([Compose]; “”; “Document”);

The macro goes to the field Body:

@PostedCommand([EditGotoField]; “Body”);

It pastes the doclink into the RichText field:

@PostedCommand([EditPaste]);

It positions the cursor back at the top entry field:

@PostedCommand([EditTop])

Testing the Formula

To test the formula:

1. Select the view All Documents.
2. Click on one of the documents to highlight it.
3. Click on the Link Documents button on the Action Bar. The document is opened for creation, the doclink is pasted, and the cursor is positioned in the first entry field.
4. Double-click on the DocLink to open the document which was selected in the view.

Properties of Actions and the Action Bar

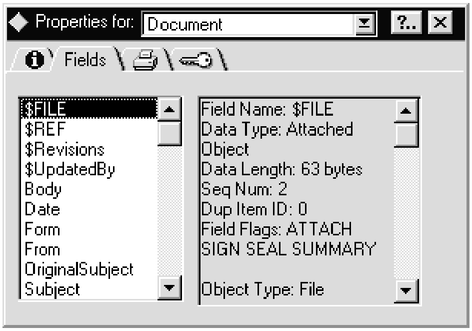
Actions and Action Bars have properties that you can display by selecting the action in the Action pane of the view or folder design window.

The properties are identical to the ones found in the Form Action Bar.

Looking at the Properties of Documents

You can look at the properties of documents when a view is displayed.

1. Select any one of the documents displayed.
2. Click on the Properties SmartIcon to display the InfoBox.
3. Click on the Fields tab to see the list of fields for that document and their values.
4. When files are attached into a document, a field called $FILE exists. Scroll down the right list box to see the file information. Here you can see the file name and size and the platform on which it was created.



$FILE
$REF
$Revisions
$UpdatedBy
Body
Date
Form
From
OriginalSubject
Subject

Field Name: $FILE
Data Type: Attached Object
Data Length: 63 bytes
Seq Num: 2
Dup Item ID: 0
Field Flags: ATTACH SIGN SEAL SUMMARY

Object Type: File

There are other keywords, such as $Revision, $Links, $UpdatedBy, and $Anonymous.

Note As you can see, all the reserved fields start with a **$** sign: Make sure not to prefix any of the fields you create with this character.

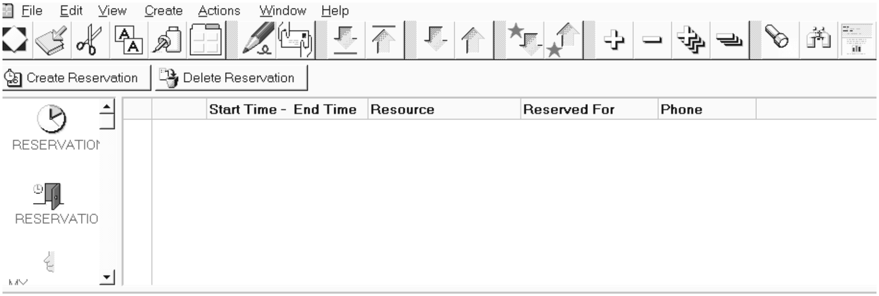
The field replication enhancement of Release 4 allows for a faster transfer of information across servers or between the servers and workstations. A new indicator is now attached to each field in all documents, which is Seq Num (or sequence number). If you have a replica of a server database, compare the values of the sequence number for fields of a replicated document. If their values are different, this means that the field containing the lowest value will be modified at the next replication.

Tip To benefit from field replication, once you are finished developing your form, create a document using this form and check the values of this indicator for all fields in the document. It will help you separate fields that are frequently updated (and replicated) from the ones that are not. This could have a major impact on the replication throughput, especially if some of the fields contain large volumes of information such as graphics, large attached files or multimedia objects (video or sound).

Using the Navigator

One of the most exciting new enhancements in Lotus Notes Release 4 is the navigator. Navigators allow the user to easily access views, Notes data, or other applications. Navigators enable the developer to create a highly graphical user interface without using an external tool such as Visual Basic.

The following figure shows an example of a navigator:



Navigators can contain text, graphics, and images. When these objects are selected, they call other Notes functions, which can do one of the following:

- Display a different view hierarchy.
- Display a different document.
- Run an @Function formula.
- Run a LotusScript.
- Launch an external application.

Navigator Objects

Navigators can contain:

- Graphic objects that you create in another program and paste into a navigator either as graphic backgrounds or as graphic buttons.

  Create a graphic background if you plan to use the picture for display only or as the foundation for navigator objects. Create graphic buttons if you want an action to occur when users click the button.

- Objects that you draw using the following navigator drawing tools:
  - Rectangle (rounded)
  - Polygon
  - Polyline
  - Ellipse
  - Textbox
  - Button
  - Hotspot polygon
  - Hotspot rectangle

  You can create the objects in the Navigator Builder. Two hotspot objects allow you to create either transparent rectangles or transparent polygons that are visible only when touched or clicked by the user. These are especially useful when you want to attach actions to different parts of a graphic background.

Adding an Action to a Navigator

You can add actions to all navigator objects except to those pasted as graphic backgrounds.

Notes provides simple actions that are easy to create and do not require any programming knowledge. A navigator can perform one of the following functions:

- Open another navigator. Clicking the object brings up the selected navigator.
- Open a view. Clicking the object brings up the selected view in the view pane.
- Serve as an alias for a folder. Clicking the object displays the contents of the designated folder in the view pane.
- Open a link. Clicking the object opens the database, view, or document link you pasted here.

In addition, a navigator can perform the following functions:

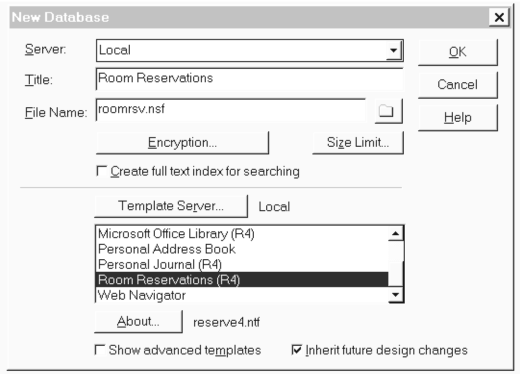
- It can run an @Function formula. This requires knowledge of the Notes macro language, but offers more choices than the simple actions supplied by Notes. Clicking the object runs the formula associated with the object.
- It can run a LotusScript program. This is a more complex function to create, but offers the most flexibility. LotusScript programs can perform tasks that are not possible with @Function formulas, such as the ability to manipulate a database access control list (ACL). Clicking the object runs the LotusScript program associated with the object.

A Navigator Example

So let's look at a simple example of a Navigator.

1. Create a new database by choosing File-Database-New from your Notes workspace.
2. In the New Database window, click the Room Reservations (R 4) template in the window listing the available templates.
3. In the Title field, type a title for the new database, for example, Room Reservations.
4. In the File Name field, type a file name for the new database, for example, roomrsv.nsf.

   The following figure shows the completed New Database window:



Server: Local

Title: Room Reservations

File Name: roomrsv.nsf

☐ Create full text index for searching

Template Server... Local

Microsoft Office Library (R4)
Personal Address Book
Personal Journal (R4)
Room Reservations (R4)
Web Navigator

reserve4.ntf

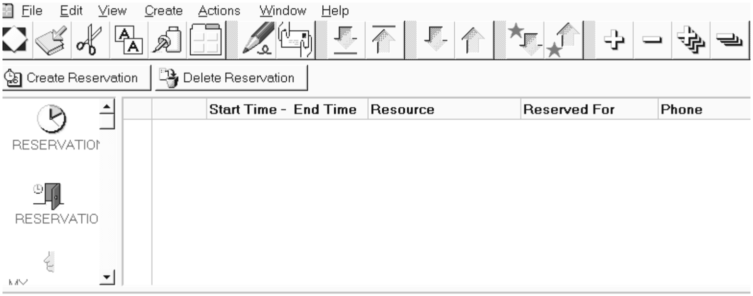
☐ Show advanced templates ☑ Inherit future design changes

5. Click OK.
6. Press the **ESC** key to leave the information window.

   The icon for the new database is added to your Notes workspace.

   You can see a two-pane window.

7. Select View-Document Preview to get a three-pane window. It looks like this:



8. Click one of the objects in the navigation pane. Notice how the view-browsing pane changes on the right-hand side. In this database, all hotspots are connected to a simple action, which is to open a view. However, you can add other objects and actions to navigator objects, as mentioned before.
9. Close the database by pressing **ESC**.

Creating a Navigator

In the following, we are going to use the Room Reservation template. It allows workgroups to schedule and reserve resources such as conference rooms or office equipment. This template is easy to modify, so this should be an easy-to-follow introduction to navigators.

We will create graphic objects as background and as a button. The button has an action associated with it.

Copying a Navigator

You can add a custom navigator to your database in one of three ways. You can:

- Copy an existing navigator from the same database.
- Copy an existing navigator from another database.
- Create a new navigator.

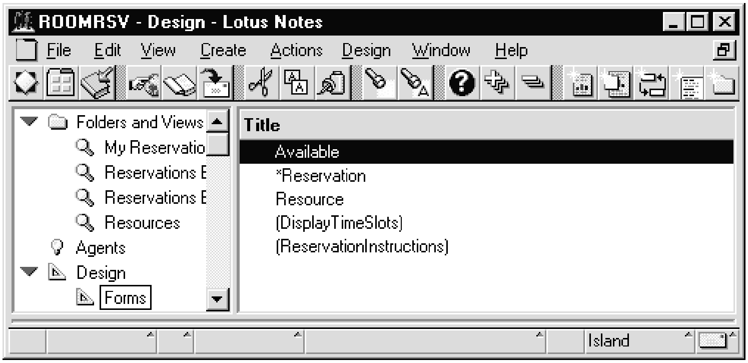
Whichever way you choose, you need designer access or higher to the database.

In our example, we will copy an existing Main Navigator to a New Main Navigator and then modify the new navigator.

To copy a navigator:

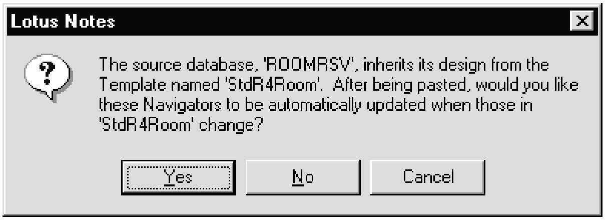
1. Select the Room Reservations database and choose View - Design.

   The following window is shown:



Title
Available
*Reservation
Resource
(DisplayTimeSlots)
(ReservationInstructions)

2. In the navigation pane, click Navigators under Design.
3. Click the navigator you want to copy in the Navigators list. In our example, only one navigator is listed. It is called Main Navigator.
4. Choose Edit - Copy.
5. Choose Edit - Paste. A message is displayed:



The source database, 'ROOMRSV', inherits its design from the Template named 'StdR4Room'. After being pasted, would you like these Navigators to be automatically updated when those in 'StdR4Room' change?

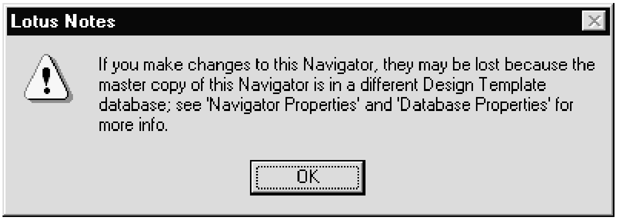
6. Click Yes to confirm.

   The view pane now lists two navigators: The Main Navigator and a copy of the Main Navigator.

Renaming a Navigator

You can rename the new navigator by following these steps:

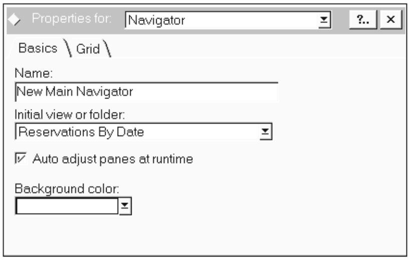
1. Double-click the new navigator listed in the right pane. The following message is displayed:



Lotus Notes

If you make changes to this Navigator, they may be lost because the master copy of this Navigator is in a different Design Template database; see 'Navigator Properties' and 'Database Properties' for more info.

2. Click OK.
3. Choose Design - Navigator Properties to display the InfoBox.
4. In the Name field, type a name for the new navigator, for example, New Main Navigator. The InfoBox looks like this:



Properties for: Navigator

Name: New Main Navigator

Initial view or folder: Reservations By Date

☑ Auto adjust panes at runtime

Background color:

   The InfoBox also allows you to change other properties, such as the initial view or folder, or the background color.

5. Close the InfoBox.

Creating Graphic Objects

We will now create graphic objects as background and as a button.

Creating a Graphic Background

First, create a graphic background.

1. If you do not have a graphic object available, use any graphic editor to create a simple graphic object. Or use any graphic object you might have.
2. Choose Edit - Copy to copy the graphic object to the clipboard.

3. The Room Reservations database should still be open, with the three panes displayed. Choose Create - Graphic Background. The graphic object is pasted into the background.

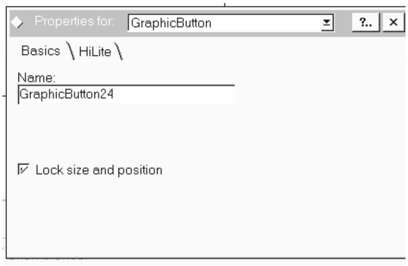
   Tip You can only have one graphic background in one navigator. You cannot move the graphic background once it has been pasted. Therefore, when you create the graphic object, consider the position and size of its components before you paste it as a graphic background.

   Tip To remove a graphic background, choose Design - Remove Graphic Background.

Creating a Graphic Button

Next, try a graphic button.

1. If you do not have a graphic object available, use any graphic editor to create a simple graphic object. Or use any graphic object you might have.
2. Choose Edit - Copy to copy the graphic object to the clipboard.
3. The Room Reservations database should still be open, with the three panes displayed. Choose Create - Graphic Button. The object is pasted as a button.
4. Move the graphic object to any position on the window by dragging it.
5. Choose Design - Object Properties to display the InfoBox.
6. Check the Lock size and position checkbox. The InfoBox now looks like this:



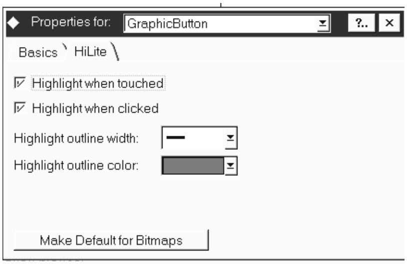
Properties for: GraphicButton

Name: GraphicButton24

☑ Lock size and position

7. Select the HiLite tab.

8. Check the Highlight when touched and Highlight when clicked check boxes. The InfoBox now looks like this:



Properties for: GraphicButton

Basics HiLite

☑ Highlight when touched

☑ Highlight when clicked

Highlight outline width:

Highlight outline color:

9. Close the InfoBox.

   Tip You can remove a graphic button by selecting it and pressing the Delete key on your keyboard.

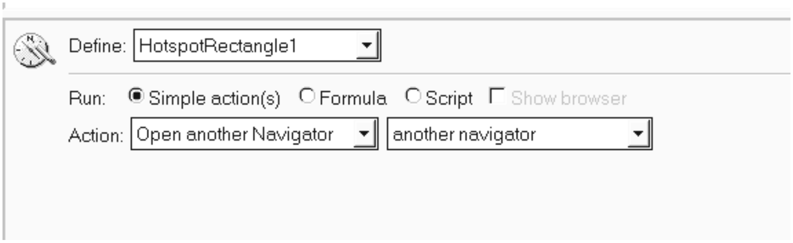
Adding an Action to a Navigator Object

You can add actions to navigator objects. It is very easy to add a simple action to an object. For example, if you want to add a simple action that opens another navigator, follow these steps:

1. Create another navigator by choosing Create - Design - Navigator.
2. As an example, choose Create - Button to create a button object.
3. Draw a box in the empty window. The InfoBox for the button is displayed.
4. In the Name field, type a name for the button.
5. In the Caption box, type some descriptive text for the button. Three additional tabs are available to specify more button features, such as font and button face color.
6. Close the InfoBox. You can see the box displayed, with the caption text showing inside.
7. Choose File - Close.
8. Click Yes to save the new navigator.
9. Assign a name to the new navigator. You should now see it listed in the view pane.
10. Open the new navigator.
11. Choose Create - Hotspot Rectangle, for example.
12. Draw a rectangle in the window. The InfoBox opens. Notice that the Define field in the bottom pane already shows HotspotRectangle1. Also, the simple action option button is selected.

13. From the Action drop-down list, choose Open another Navigator.

14. From the drop-down list next to the Action drop-down list, choose another navigator. It now looks like this:



Define: HotspotRectangle1

Run: ● Simple action(s) ○ Formula ○ Script □ Show browser

Action: Open another Navigator | another navigator

15. Choose File - Save.

16. Close the InfoBox.

17. Press **ESC** and confirm that you want to save your changes.

Creating a Text Box

Next, create a text box for the new hotspot.

1. Open the new navigator.
2. Choose Create - Text.
3. Draw a box anywhere in the window, close to the hotspot. The InfoBox opens.
4. In the Name field, type a name for the text box, for example, text1.
5. In the Caption box, type a caption describing the text box.
6. If required, check the Lock size and position check box.
7. Close the InfoBox. You should see the text box, with the caption text displayed inside.
8. Press **ESC** to leave the window and confirm that you want to save your changes.

Adding an Action Using LotusScript

If you require a more complicated action to be added to a graphic object, you can create the action by using an @Function or a LotusScript program.

To add an action using LotusScript, follow these steps:

1. Select a graphic object.
2. Choose the Script option button.
3. Make sure that Click is selected in the Event area. This ensures that the LotusScript program is run when the user clicks the object.

4. In the edit box, type the following:

```
Sub Click(Source As Navigator)
Dim db As New NotesDatabase("", "ROOMRSV.NSF")
Dim view As NotesView
Dim doc As NotesDocument
Dim capacity, room As NotesItem
Dim people As Integer
Dim msg As String
people = Cint( Inputbox( "How many people are
             estimated to attend the meeting ? " ) )
Set  view = db.GetView( "Resources")
Set  doc =  view.GetFirstDocument
While Not ( doc Is Nothing )
  Set capacity = doc.GetFirstItem("Capacity")
  Set room = doc.GetFirstItem("ResourceName")
  If ( Cint(capacity.text ) => people ) Then
    msg = msg + room.text + "(" + capacity.text + ") "
  End If
  Set  doc =  view.GetNextDocument( doc )
Wend
If msg = "" Then
  Messagebox( "No room meets your request." )
Else
  Messagebox( msg + " meet your request." )
End If
End Sub
```

Note Notes compiles the script when you close it. Compiling is the process by which a script is translated into executable code.

Testing a Navigator

To test a newly designed navigator, follow these steps:

1. Open the new navigator.
2. Choose Design - Test Navigator.
3. Highlight and click each object to see if the highlighting and the actions are as expected.
4. If the test is not satisfactory, choose Design - Test Navigator again to return to design mode and make changes as required.
5. Next, test the navigator using some documents. For navigators whose actions perform multiple steps or complex tasks, split the process into several smaller tasks and create an action for each task. Test and fix each small task first. When everything is working correctly, combine the formulas into one, and then test the navigator again.

Including a Navigator in the View Menu

To display a navigator in the View - Show submenu when a database is opened, follow these steps:

1. Open the database that contains the navigator.
2. Choose File - Database - Properties.
3. Click the Launch tab.
4. To display the navigator in the navigation pane, choose Open designated navigator under On Database Open. If you want to display the navigator in a full-screen window, select Open designated navigator in its own window.
5. From the Navigator drop-down list, select the navigator you want displayed in the view.
6. Close the InfoBox.
7. Choose File - Close.

   When the database is opened, the View - Show submenu should now list the navigator.

Chapter 6
Programming in Lotus Notes

There are a number of different ways of creating Notes applications, both within Notes and outside of Notes. Within Notes you can create new applications by:

- Setting up a new database using one of the application templates shipped with Notes, and using it as-is, or customizing it for your own use. The templates shipped with Notes use a wide variety of Notes features, and may be used as examples of how to write your own applications. They are described briefly below.
- Creating a new application from scratch using the Blank database template, and then developing your own forms and views.

When programming in Notes, you can use any of the following:

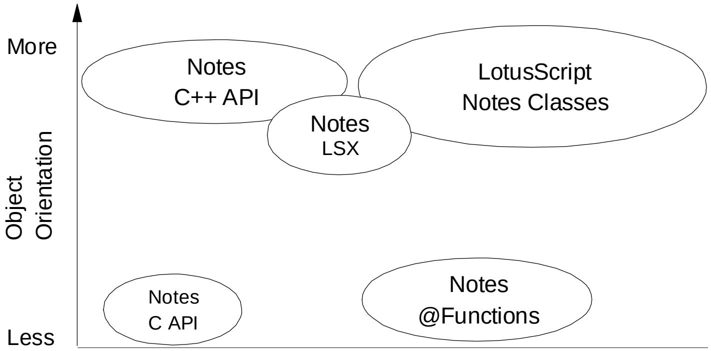
- Simple Actions
- Formulas
- LotusScript
- LSXs (LotusScript Extensions)

In addition, it is possible to develop Notes applications by programming using External Development Tools such as:

- Lotus Notes C API
- Lotus Notes C++ API
- Interflox API for REXX (OS/2 only)

Your choice of development tools will depend on the application, the skills and experience of your developers, and the environment in which you are working.

Notes Release 4 Programmability Map



If you have a professional development environment already, you may be more likely to leverage existing skills and use the C/C++ or Visual Basic APIs. Corporate developers and advanced Notes users are more likely to use built-in Notes tools such as @Functions and LotusScript.

Templates

The following section lists some of the application templates supplied with Notes. You may use them as-is, or customize them to your needs.

Approval Cycle

This database provides a place from which organizations can manage their electronic approvals. Using the ApprovalLogic subform, a different form can be designed for each type of approval while all approval forms use the same approval logic. Consequently, when an organization changes its approval policies, only the ApprovalLogic subform will need to be changed, allowing for fast response to changing business needs.

Designers will derive several benefits from using this database:

- They will not need to rewrite approval logic for each approval form because it is already provided in a subform.
- The approval logic provided on the subform is extremely flexible. Because all approval situations are not identical, the designer will fill out an application profile for each approval form, which tells the ApprovalLogic subform how this particular approval should be processed.

- When organizational approval policies change, the designer does not need to change every form. Instead they can either edit the application profile documents or, if necessary, they can modify the ApprovalLogic subform.

Discussion

This template creates an electronic conference room used by workgroups to share their thoughts and ideas. The template includes archiving capabilities and an “Interest Profile” that allows users to automatically mail themselves document links to topics of interest. Also of interest to developers is the ability to migrate a Notes Release 3 version of the database.

Document Library

A Document Library application is an electronic filing cabinet that stores reference documents for access by a workgroup. The database might contain anything from environmental impact statements for a group of engineers to financial statements for a group of loan officers.

In this database, developers will find examples of:

- Review Cycle: Used to route a document to a series of recipients.
- Document Archiving: Used to move expired documents to an archive database.

The template is also available for specific application suites, listed below. These templates automatically launch, store, and support review cycles of documents created with the following suite products that include word processing and spreadsheet applications:

- SmartSuite Library
- Microsoft Office Library

Note Databases created with this template can only launch the objects on the Windows platforms. Objects can be viewed on all platforms.

Personal Journal

There are two basic types of documents in the Personal Journal template — a Journal Entry and a Clean Sheet. The main difference is that the Journal Entry has a Title field which is displayed in the view, whereas the Clean Sheet does not.

When you have finished creating a Clean Sheet, choose File - Save and you will be prompted for the Title of the entry. If you ever need to change the Title of a clean sheet document — just choose the “Doc Info...” button in the document.

Other features of interest to the developer include:

- The “All Documents” view shows all documents that have been created in the personal journal. You can click on the Title column heading to sort alphabetically by title and then click again to restore the View to its original state. Similarly, you can click on the date column to get the documents in descending creation order.
- If you are writing a long document and want to keep multiple versions, choose the Save As New Version menu item from the File menu when you are editing the document. By doing this you will be a saving a “hierarchy” of documents with the newest version on top and the older versions underneath. Any time you move the document to a folder, all its prior versions will go along with it.
- You can rename a document without opening the document from any of the Views or Folders by choosing the Doc Info... button in the View or folder.

Room Reservations

Reservation Scheduler is an application designed to allow workgroups to schedule and reserve physical resources such as conference rooms or office equipment.

InterNotes Web Navigator

The InterNotes Web Navigator is a Notes Release 4 feature that allows you to navigate through pages on the Web directly from your Notes environment. The Web Navigator is much more than a Web browser — it combines the features of a Web browser with the powerful capabilities of Lotus Notes.

Developers should look at this template as a good example of the use of Navigators.

Shared Template Components

This template includes components which are used in multiple other templates. It is not intended to be used in creating new databases. When a design refresh is performed on some of the databases created from other templates, elements of this template may be pulled in. For example, the design elements used for archiving are refreshed from this template, into the Discussion template.

Items of interest to developers are as follows:

Forms

Archive Profile (indicates archive parameters)

Archive Log (lists docs that were archived during each run)

Views
Archive Logs

($Profiles)

Agents
Edit Archive Profile (create / edit the Archive Profile doc)

Mark Document as Expired (toggle ExpireDate on / off)

Periodic Archive (background agent - picks up inactive and expired docs)

Archive Selected Documents (archives selected docs)

ACL Roles
ArchiveManager (is allowed to edit Archive Profile, and see Archive Logs)

Programming in Notes

The following section details the differences between the three integral programming interfaces to Notes. It will give you a short overview of Notes' Simple Actions, and it will compare LotusScript and @Functions.

Simple Actions

These are predefined actions which allow you to define a sequence of actions without requiring any programming knowledge. They are ideal for the end user who wishes to automate some routine tasks. In addition, they provide significant power to the developer and eliminate manual coding. The simple actions available are:

Copy to database

Copy to folder

Delete from database

Mark document read

Mark document unread

Modify field

Modify fields by form

Move to folder

Remove from folder

Reply to sender

Run agent

Send document

Send mail message

Send newsletter summary

Formulas

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

@Functions are a powerful tool when you want to manipulate the current Notes document in an application, since the developer need not obtain the context for the document.

LotusScript

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

Furthermore, the hierarchy of the Notes Classes represents the flow of control you follow in the user interface if you step down from a database icon to a view, and further on to a document and to a specific field within this document. For example, if you are coding in LotusScript you will start with the UIWorkspace class and go down to the UIDocument class which represents the currently open document. Once you have set this object variable you have access to the fields of the document. The same principle applies if you are working with the back-end classes of Lotus Notes, which represent the objects you might wish to work with that are not in the user interface. You will start at the NotesSession class and go down through the NotesDatabase class to the NotesDocument class.

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

- **Superset of BASIC**

  Since LotusScript is a superset of the BASIC language, it is easy to learn, especially for Visual Basic users. You can write sophisticated scripts by using conditions, branches, subroutines, while loops, and others.

- **Cross-platform**

  LotusScript is a multi-platform BASIC-like scripting language. Major platforms are supported, such as Windows, Macintosh, OS/2 and UNIX. You can create just one application, which can be used on any platform.

- **Object-oriented**

  Notes Release 4 provides Notes Object Classes that are available to LotusScript. You can write scripts to access and manipulate these objects. The scripts are event-driven, such as by an action, clicking the object or button, opening a document, or opening a view.

- **Included in Lotus Applications**

  Since LotusScript is supported by all the Lotus products, these products are able to access Notes classes using a Notes-supplied LotusScript extension. Another advantage is that you only need to learn one language to become proficient in writing scripts in other Lotus products.

- **OLE Support**

  Notes can be the perfect container for SmartSuite documents and other OLE-enabled applications, such as Microsoft Office. You can use external OLE 2.0 automation objects by scripting them, such as 1-2-3 worksheet objects.

  Notes registers itself as an OLE automation server. External applications can use these objects in scripts to create and reference them.

  LotusScript is able to combine all the parts and provide the means for controlling and manipulating objects.

- **Coexistence with Notes @Functions**

  Lotus continues to support @Functions. LotusScript can work with them.

- **Integrated Development Environment**

  The LotusScript Integrated Development Environment (IDE) provides an easy-to-use interface to create, edit, and debug scripts, and to browse variables and properties of classes. This allows you to write more complex scripts in Notes.

- **Extendable through LSXs**

  You may extend LotusScript by writing your own classes, which are called LotusScript extensions (LSXs). Creating your own LSXs allows you to expose custom functionality to LotusScript developers in precisely the same way as Notes functionality is exposed. You might use this, for example, if you have customer processing logic, such as a proprietary pricing process, you wanted to make available to Notes developers.

Using LotusScript Notes Classes

This section describes the LotusScript Notes classes. Notes provides Object Classes for Notes that are available to LotusScript so that you can manipulate Notes Objects using Notes functions in your script. You will see how to manipulate data in Notes and which actions to take to make your changes effective and consistent.

If you would like to know the method, property or class in detail, refer to the appropriate chapter in the *Programmer's Guide* for Notes Release 4.

Notes Classes

Two types of Notes object classes are provided:

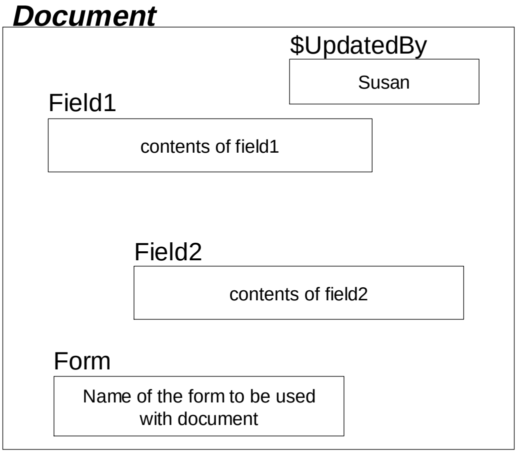
- Front-end UI (user interface) classes
- Back-end classes

Understanding Front-End and Back-End Classes

First of all you need to consider how data are stored in Notes. You can think of a document within a Notes database as a record, but a Notes document is more sophisticated than a typical database record. It may contain rich text, pictures, objects, and many other types of information. For example, if you access a Notes document using the back-end classes you may manipulate the contents of the fields, may add new fields to the document, or remove fields from the document. However, if you make such changes the input translation and input validation formulas contained in a form are not executed.

On the other hand, if you work with Notes' front-end classes your changes to the fields are visible to the user. For example, if you invoke the Refresh method of the NotesUIDocument class the input translation and input validation formulas are carried out.

The following picture represents a back-end document and shows how data is stored in a Notes database:



The fields *Field1* and *Field2* have been defined in the form which was used to create this document. The name of this form is stored in the field *Form*. If you change the value of the field *Form* using an agent or LotusScript the document will be presented to the user using the other form when it is opened next time.

Note If there is no form field within a document Notes will display such a document using the database's default form. If there is no default form, the document cannot be displayed.

Field *$UpdatedBy* has been created by Notes and contains a list of users who have worked on this document.

Note Field names starting with $ are used and maintained by Notes.

Front-End UI Classes

UI classes provide features to emulate user actions. They also provide access to objects such as Workspace, Database window, Document window, Field, and Rich-Text Field.

- **NotesUIWorkSpace**

  Represents the current Notes workspace window.
- **NotesUIDatabase**

  Represents the currently used database.

- **NotesUIView**

  Represents the currently used view.

- **NotesUIDocument**

  Represents the document that is currently open.

The following Notes classes have only events associated with them.

- **Button**

  Objects of this class represent a button.

- **Field**

  Objects of this class represent a field.

- **Navigator**

  Objects of this class represent a navigator.

Back-End Classes

Back-end classes represent Notes objects such as Database, View, Agent, Document, Item. You can use these classes to manipulate Notes elements directly in LotusScript. The following back-end classes exist.

- **NotesSession**

  Represents the Notes environment of the current script, providing access to environment variables, Name & Address Books, information about the current user, and information about the current Notes platform and release number.

- **NotesDbDirectory**

  Represents the Notes databases on a specific server or local machine.

- **NotesDatabase**

  Represents a Notes database.

- **NotesACL**

  Represents the Access Control List (ACL) of a database.

- **NotesACLEntry**

  Represents a single entry in an Access Control List. An entry may be for a person, a group, or a server.

- **NotesAgent**

  Represents an agent.

- **NotesView**

  Represents a view or folder of a database and provides access to documents within it.

- **NotesViewColumn**

  Represents a column in a view or folder.

- **NotesDocumentCollection**

  Represents a collection of documents from a database, selected according to specific criteria.

- **NotesDocument**

  Represents a document in a database.

- **NotesItem**

  Represents a piece of data in a document. All of the items in a document are accessible through LotusScript, regardless of what form is used to display the document in the user interface.

- **NotesRichTextItem**

  Represents an item of type rich text.

- **NotesEmbeddedObject**

  Represents embedded objects, linked objects, and file attachments.

- **NotesDateTime**

  Represents a date and time. Provides a means of translating between the LotusScript date-time format and the Notes format.

- **NotesDateRange**

  Contains a range of NotesDateTime. An object of type NotesDateTime represents a given date and time.

- **NotesLog**

  Enables you to record actions and errors that take place during a script's execution. You can record actions and errors in a Notes database, a mail memo, or a file (for scripts that run locally).

- **NotesNewsLetter**

  Represents a document that contains information from, or doclinks to several other documents. All of the NotesItem properties and methods can be used on a NotesRichTextItem, too.

- **NotesForm**

  Represents a form in a Notes database.

- **NotesInternational**

  This class contains properties which provide information about the international settings, for example date format, of the environment Notes is running in.

- **NotesName**

  Properties of this class contain information about a Notes user name.

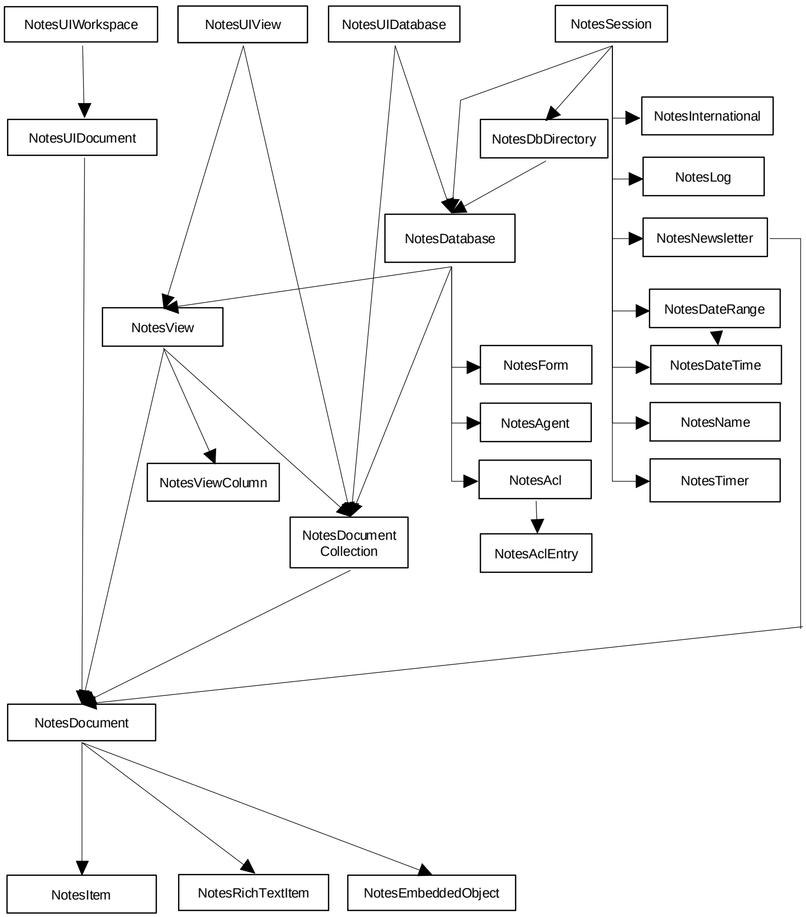
- **NotesTimer**

  Objects of this class represent a timer in Notes.

Class Hierarchy

There is a hierarchical relationship for object classes. Higher hierarchical classes contain the lower ones. The class hierarchy looks like this:

LotusScript Classes



Each class has defined members, properties and methods. Using these members, you can access other objects. The relationship of containment and access means that the higher class has the property or the method to access the lower one.

For example, you can see all the views when you open the database. This means that the opened database(object) in the workspace includes the views(object). Furthermore, you can see the documents when you select one of the views. This means that your selected view(object) contains the documents(object). This hierarchy is important when using Notes classes.

Let's look at some examples of code which use classes.

Example 1: Getting the text of the Subject field

```
Dim session As New NotesSession
Dim db      As NotesDatabase
Dim view    As NotesView
Dim doc     As NotesDocument
Dim item    As NotesItem
Set db   = session.CurrentDatabase
Set view = db.GetView( "Main View" )
Set doc  = view.GetFirstDocument
Set item = doc.GetFirstItem( "Subject" )
```

First we declare the variable *session* as types of *NotesSession* class, and *New* is used to create an instance of that class.

Variables *db, view, doc, item* we declare as types of *NotesDatabase, NotesView, NotesDocument, NotesItem* classes respectively.

To get the text of the subject field, we need to follow the hierarchical path from the top to the lower one. In this example we go from *NotesSession* class to *NotesItem* class:

NotesSession - NotesDatabase - NotesView - NotesDocument - NotesItem.

We initialize the object reference variable *db* with the property *CurrentDatabase* of the higher level class.

Then we set the object variable *view* using the *GetView* method.

The next statements are the same as before, we use the methods *GetFirstDocument* and *GetFirstItem* to get the objects in the lower hierarchical class.

Example 2: Disabling a role for a person

```
Dim session As New NotesSession
Dim db      As NotesDatabase
Dim acl     As NotesACL
Dim entry   As NotesACLEntry
Set db    = session.CurrentDatabase
Set acl   = db.ACL
Set entry = acl.GetEntry("Gina Widmann")
Call entry.DisableRole("Auditor")
Call acl.Save
```

To access the personal ACL data, you need to follow the hierarchical path from the top to the lower one. This example steps from the *NotesSession* class to the *NotesACLEntry* class:

NotesSession - NotesDatabase - NotesACL - NotesACLEntry.

The object that you would like to manipulate has methods or properties to handle its own data. The first seven lines of this example are similar to Example 1. The eighth line uses the *DisableRole* method of the *NotesACLEntry* class to disable the role ("Auditor" for a person) "Gina Widmann."

Example3: Getting the subject field of all documents

```
Dim db    As New NotesDatabase("Server","db.nsf")
Dim dc    As NotesDocumentCollection
Dim doc   As NotesDocument
Dim item  As NotesItem
Dim subject As String
Set dc = db.AllDocuments
Set doc = dc.GetFirstDocument()
While Not(doc Is Nothing)
  Set item =  doc.GetFirstItem("Subject")
  subject = item.text
  Set doc = dc.GetNextDocument(doc)
Wend
```

The earlier two examples start at the *NotesSession* class, but to access an existing database when you know its server and file name, you can get the database object directly as shown in the first line. This illustrates a unique feature of writing Notes applications in LotusScript as opposed to the formula language: You can access any database from within a script. The following sequence is the same as in the earlier examples. The *NotesDatabase* class contains the *NotesDocumentCollection* class, which contains *NotesDocument*:

NotesDatabase - NotesDocumentCollection - NotesDocument - NotesItem.

We use the *AllDocuments* property of the *NotesDatabase* class to get all the documents in the database.

Next, we use the *GetFirstDocument* method of the *NotesDocumentCollection* class to get the first document in a collection.

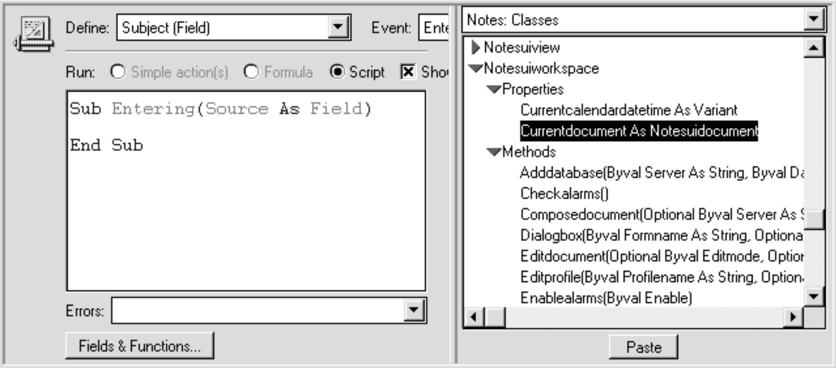
Then we use the *GetNextDocument* method of the *NotesDocumentCollection* class to get the document immediately following the earlier document in a collection. If a document does not exist in a collection, the *GetNextDocument* method returns *Nothing.*

Using the Object Browser

Notes provides an object browser you can use when working with LotusScript Notes classes.

To display the browser:

1. Go to the design mode.
2. Select an object and click on the Script button in the programmer pane.
3. Click on the Show Browser button.



Define: Subject (Field)

Event: Ente

Run: ○ Simple action(s) ○ Formula ● Script ☒ Sho

```
Sub Entering(Source As Field)
End Sub
```

Errors:

Notes: Classes

```
▶Notesuiview
▼Notesuiworkspace
  ▼Properties
      Currentcalendardatetime As Variant
      Currentdocument As Notesuidocument
  ▼Methods
      Adddatabase(Byval Server As String, Byval Da
      Checkalarms()
      Composedocument(Optional Byval Server As S
      Dialogbox(Byval Formname As String, Optiona
      Editdocument(Optional Byval Editmode, Optior
      Editprofile(Byval Profilename As String, Option
      Enablealarms(Byval Enable)
```

You can select the following parts from the browser drop-down listbox:

- LotusScript Language
- Notes: Classes
- Notes: Constants
- Notes: Subs and Functions
- Notes: Variables
- OLE Classes

Although you can list the LotusScript built-in functions, as well as all the classes and their methods and properties in the browser listbox, you do have to know the syntax of their usage. The browser does not cover the syntax in any detail. However, the browser will help you write the script. You do not need to check the manual every time. And you can also get context-sensitive help by selecting the property, method, or event you are interested in and pressing the **F1** key.

Note You will need the full Notes help file on your disk if you are not connected to a network.

For example, the browser is useful in the following situations:

- You are not sure of the spelling of the functions, methods and properties.
- You are not sure of the relationships of the Notes classes.
- You would like to confirm which constants are available.
- You are not sure of the type of the return value from functions or methods.
- You need to know what functions are available.

Event Programming With LotusScript

When you program in Lotus Notes you add your LotusScript code to Notes objects. Your code is executed by the occurrence of an event to the objects such as click a button, open a document, close a document, or entering data in a field.

You can write a very simple script for an object such as a button, for example:

```
Sub Click( Source As Button )
   MessageBox( "Welcome to LotusScript" )
End Sub
```

This script just shows a message box when you click the button.

Programmable Objects

The following table outlines the programmable objects in Notes. The third column specifies whether the object supports scripts, formulas, or both. Before you write a script or formula, make sure that a simple action won't do the task.

Scope	*Notes object*	*Type*
Workspace	SmartIcons®	Formula
Database	Replication formula	Formula
	Agent	Both
View design	Form formula	Formula
	Selection formula	Formula
	Column formula	Formula
	Action	Both
	Hide action formula	Formula
Form design	Window title formula	Formula
	Section title formula	Formula
	Section access formula	Formula
	Insert subform formula	Formula
	Hide paragraph formula	Formula
	Action	Both
	Hide action formula	Formula
	Event	Both
	Button	Both
	Hotspot	Both
Navigator design	Hotspot	Both
Layout region design	Hotspot	Both
Field design	Default value formula for editable field	Formula
	Input translation formula for editable field	Formula
	Input validation formula for editable field	Formula
	Value formula for computed field	Formula
	Keyword field formula	Formula
	Event	Script
Rich text field	Button	Both
	Hotspot	Both
	Section title formula	Formula
	Hide paragraph	Formula

Events

Some Lotus Notes objects have events associated with them. The following is an overview of which events are supported for a specific object.

	Database	*View*	*Form*	*Field*	*Button*	*Action*	*Agent*
Click					x	x	
Entering				x			
Exiting				x			
Initialize	x	x	x	x	x	x	x
Objectexecute					x	x	
Postdocumentdelete	x						
Postdragdrop		x					
Postmodechange			x				
Postopen	x	x	x				
Postpaste		x					
Postrecalc			x				
Queryaddtofolder		x					
Queryclose	x	x	x				
Querydocumentdelete	x						
Querydocumentundelete	x						
Querydragdrop		x					
Querymodechange			x				
Queryopen		x	x				
Queryopendocument		x					
Querypaste		x					
Queryrecalc		x					
Querysave			x				
Regiondoubleclick		x					
Terminate	x	x	x	x	x	x	x

Event Type and Sequence

Database object

The Database object has the following events:

- Initialize (when it is being loaded)
- Postopen (after it is opened)
- Querydocumentdelete (before deleting a document)

- Querydocumentundelete (before undeleting a document)
- Postdocumentdelete (after deleting a document)
- Queryclose (before it is closed)
- Terminate (when it is being closed)

View object

The View object has the following events:

- Initialize (when it is being loaded)
- Queryopen (before it is opened)
- Postopen (after it is opened)
- Queryopendocument (before a document is opened)
- Querydragdrop (before a drag-drop operation in a calendar view)
- Pastedragdrop (after a drag-drop operation in a calendar view)
- Querypaste (before a document is pasted into the view)
- Postpaste (after a document is pasted into the view)
- Queryaddtofolder (before a document is added to a folder)
- Regiondoubleclick (after the region is double-clicked in a calendar view)
- Queryrecalc (before the view is recalculated)
- Queryclose (before the view is closed)
- Terminate (when the view is closed)

Form (document) object

The Form object has the following events:

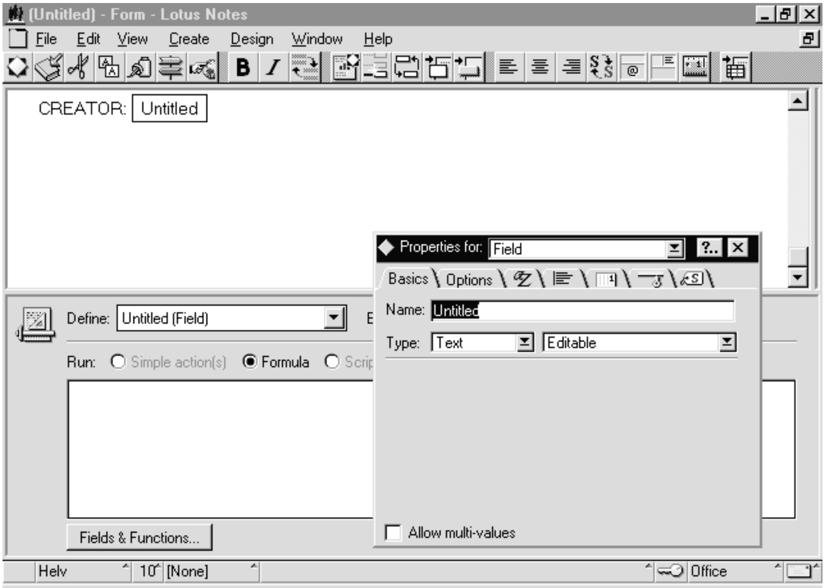
- Initialize (when it is being loaded)
- Queryopen (before it is opened)
- Postopen (after it is opened)
- Postrecalc (after it is refreshed)
- Querysave (before it is saved)
- Querymodechange (before changing to or from edit mode)
- Postmodechange (after changing to or from edit mode)
- Queryclose (before it is closed)
- Terminate (when it is being closed)

Example

Let's try to add an action to the Form (document) object and use the *Postopen* event.

Note You should try the following example by creating a temporary database based on the Blank template so as not to corrupt any existing databases.

1. Select the database and choose Create - Design - Form. The form design window is opened and the cursor is blinking at the top left-hand side.
2. Enter CREATOR: at the cursor blinking position.
3. Choose Create-Field. The InfoBox used to set the properties of a field appears.



CREATOR: Untitled

Name: Untitled

Type: Text Editable

□ Allow multi-values

Define: Untitled (Field)

Run: ○ Simple action(s) ● Formula ○ Scrip

4. Enter CREATOR in the Name: field of the InfoBox box, and close the box.
5. Choose Untitled(Form) from the Define: drop-down combo box in the programmer pane.
6. Choose Postopen from the Event: drop-down combo box in the programmer pane.

7. Edit the LotusScript so that it looks exactly like this:

```
Sub Postopen(Source As Notesuidocument)
  Dim session As New NotesSession
  If source.EditMode Then
    Call
source.FieldSetText("Creator",session.CommonUserName)
  End If
End Sub
```

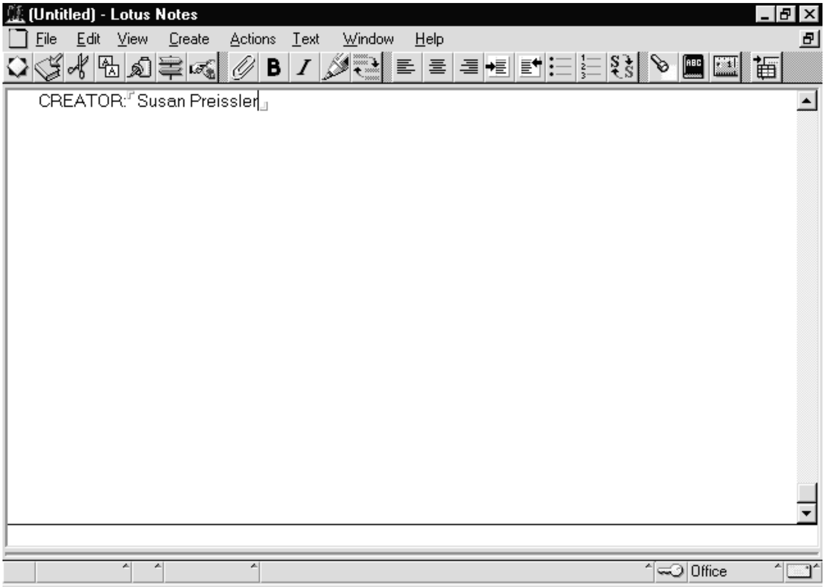
8. Choose File - Save.
9. Enter SAMPLE1 as the form name and click OK.
10. Choose File - Close.

Running the Example

1. Select the database that you added the script to.
2. Choose Create - SAMPLE1.

The new form SAMPLE1 appears and your name is set in the creator field.

This script runs after the user opens the document. If the document is new, the Creator field is set to the name of the creator. You can select any events mentioned earlier and write a script. For example, you can select the QuerySave event to check whether every field has a value entered in it or not before the document is saved.



CREATOR: Susan Preissler

Field Object

The Field object has the following events:

- Initialize (when it is being loaded)
- Entering (when it is entered in edit mode)
- Exiting (when it is exited in edit mode)
- Terminate (when it is being closed)

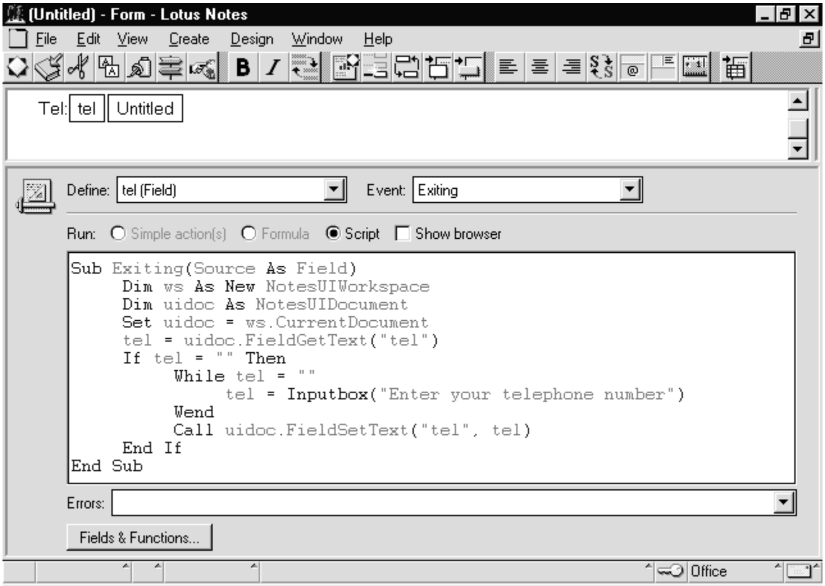
Example

Let's try to add an action to a Field object and use the *Exiting* event.

Note You should try the following example by creating a temporary database so as not to corrupt any existing databases.

1. Select the database and choose Create-Design-Form. The form design window is opened and the cursor is blinking at the top left-hand side.
2. Enter TEL: at the cursor blinking position.
3. Choose Create-Field. The InfoBox used to set the properties appears.
4. Enter TEL in the Name: field of the InfoBox box.
5. Create one more field. You don't need to change the name in the InfoBox, you can leave it as Untitled. When we later on go to this field we will exit from the TEL field which will cause the exiting event to occur.
6. Choose TEL(Field) from the Define: drop-down listbox in the programmer pane. Choose Exiting from the Event: drop-down listbox.
7. Edit the LotusScript so that it looks exactly like this:

```
Sub Exiting( Source As Field )
  Dim ws    As New NotesUIWorkspace
  Dim uidoc As NotesUIDocument
  Set uidoc = ws.CurrentDocument
  tel = uidoc.FieldGetText("TEL")
  If tel = "" Then
    While tel = ""
      tel = Inputbox("Enter your telephone number")
    Wend
    Call uidoc.FieldSetText("TEL", tel)
  End If
End Sub
```


(Untitled) - Form - Lotus Notes
File Edit View Create Design Window Help
Tel: tel Untitled
Define: tel (Field)
Event: Exiting
Run: ○ Simple action(s) ○ Formula ● Script □ Show browser
Errors:
Fields & Functions...
Office

```
Sub Exiting(Source As Field)
     Dim ws As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Set uidoc = ws.CurrentDocument
     tel = uidoc.FieldGetText("tel")
     If tel = "" Then
          While tel = ""
               tel = Inputbox("Enter your telephone number")
          Wend
          Call uidoc.FieldSetText("tel", tel)
     End If
End Sub
```

Choose File - Save. You are asked to specify a form name for the new form.

8. Enter SAMPLE2 as the form name and click on OK.
9. Choose File - Close.

Running the example

1. Select the database that you added the script to.
2. Choose Create - Sample2. The new form SAMPLE2 appears.
3. Select the second field without entering any data. A message box appears which asks you to enter your telephone number.

This script runs when the user exits from the TEL field. The script makes sure that the user enters a telephone number.

Button Object

The Button object has the following events:

- Initialize (when it is being loaded)
- Click (when it is selected)
- ObjectExecute (see note below)
- Terminate (when it is being closed)

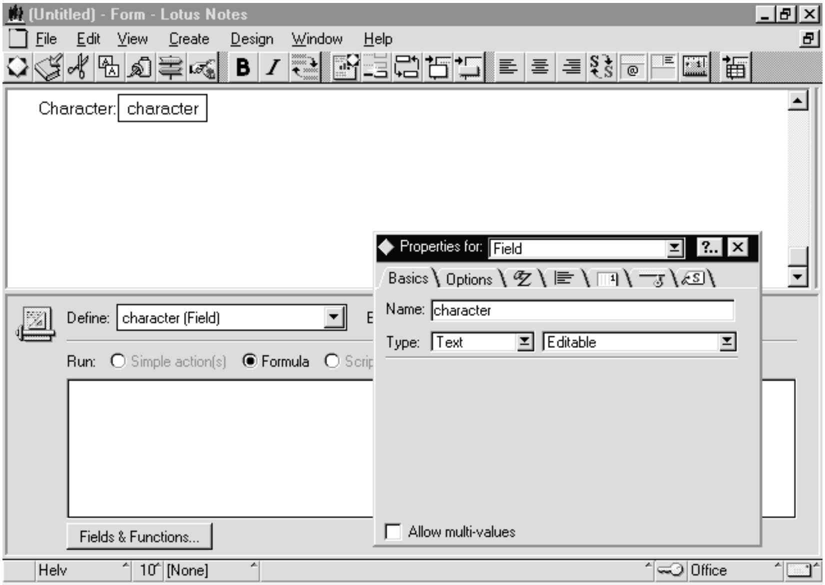
Note The ObjectExecute event is primarily used in external applications and should not be used in the Notes environment.

Example

Let’s try to add an action to a Button object and use the Click event.

Note You should try the following example by creating a temporary database so as not to corrupt any existing databases.

1. Select the database and choose Create - Design - Form. The form design window is opened and the cursor is blinking at the top left-hand side.
2. Enter Character: at the cursor blinking position.
3. Choose Create - Field. The InfoBox used to set the properties appears.
4. Enter character in the Name: edit box of the InfoBox.



Character: character

Properties for: Field

Name: character

Type: Text Editable

☐ Allow multi-values

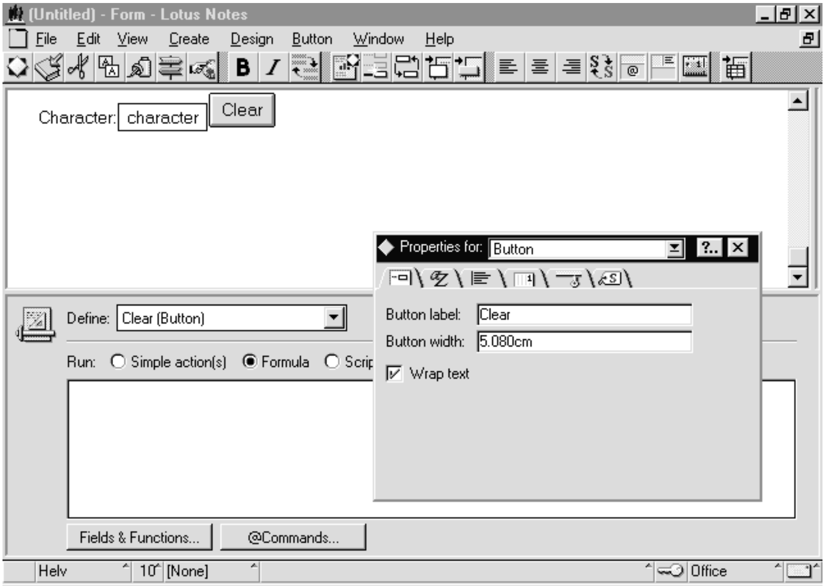
Define: character (Field)

Run: ○ Simple action(s) ● Formula ○ Scrip

Helv 10 [None] Office

5. Set the cursor position just to the right side of the character field.
6. Choose Create - Hotspot - Button. A button is placed on the form, and the InfoBox for the button appears.

7. Enter Clear in the Button label: edit box in the InfoBox.



(Untitled) - Form - Lotus Notes
File Edit View Create Design Button Window Help

Character: character

Properties for: Button

Button label: Clear

Button width: 5.080cm

☑ Wrap text

Define: Clear (Button)

Run: ○ Simple action(s) ● Formula ○ Scrip

Fields & Functions...
@Commands...
Helv
10
[None]
Office

8. Choose CLEAR(Button) from the Define: drop-down combo box in the programmer pane.

9. Choose the Script button.

10. Edit the sub so that it looks exactly like this:

```
Sub Click(Source As Button )
     Dim ws    As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Set uidoc = ws.CurrentDocument
     If ( uidoc.FieldGetText( "Character" ) <> "" ) Then
          Call uidoc.FieldClear("Character")
     End If
End Sub
```

11. Choose File - Save. You are asked to specify a form name for the new form.

12. Enter SAMPLE3 as the form name and click on OK.

13. Click on the File - Close menu.

Running the example

1. Select the database that you added the script to.

2. Choose Create - SAMPLE3. The new form SAMPLE3 appears.

3. Enter some characters in the field, then click on the Clear button. The characters you entered are cleared.

Action Object

The Action object has the following events:

- Initialize (when it is being loaded)
- Click (when it is selected)
- ObjectExecute (see note below)
- Terminate (when it is being closed)

Note The ObjectExecute event is primarily used in external applications and should not be used in the Notes environment.

Agent Object

The Agent object has the following events:

- Initialize (when it is being started)
- Terminate (before it is closed)

How Scripts and Formulas Are Executed

If your application contains a combination of LotusScript and the formula language, it's useful to know the order in which the events and formulas in a form are executed in Notes.

The following example lists the order in which LotusScript events and Notes formulas in a single form's design are executed during a number of activities. The list was generated by embedding messagebox commands or @Prompt formulas into all the possible events and formulas on a test form containing different field types. The form does not include all the possible field types or evaluation combinations. By studying the results in this example, however, you may be able to better understand the order of execution in the forms of your own application.

The test form contains five fields from top to bottom in the following order:

- Subject - Editable / Text Field - (with Default Value, Input Translation and Input Validation Formulas)
- From - Computed When Composed / Authors Name Field - (with Value Formula)
- Counter - Computed / Number Field - (with Value Formula)
- DisplayNum - Computed For Display / Number Field - (with Value Formula)
- Body - Editable / RTF Field - (with Default Value Formula)

The following tables show you different activities, such as composing a document, and the order in which the LotusScript events and Notes formulas are executed for each activity.

Composing a Document

Object	*Formula or Event*
Form	Initialize Event
Form	Window Title
Form	Query Open Event
Subject Field	Default Value Formula
Subject Field	Initialize Event
From Field	Value Formula
From Field	Initialize Event
Counter Field	Value Formula
Counter Field	Initialize Event
DisplayNum Field	Value Formula
DisplayNum Field	Initialize Event
Body Field	Value Formula
Body Field	Initialize Event
Subject Field	Entering Event
Form	PostOpen Event

Saving a Document Using @Command([FileSave]) or File-Save

Object	*Formula or Event*
Form	QuerySave Event
Subject Field	Input Translation Formula
Counter Field	Value Formula
DisplayNum Field	Value Formula
Subject Field	Input Validation Formula

Closing the Window using @Command([FileCloseWindow]) or File-Close

Object	*Formula or Event*
Form	QueryClose Event
Form	Terminate Event
Subject Field	Terminate Event
From Field	Terminate Event
Counter Field	Terminate Event
DisplayNum Field	Terminate Event
Subject Field	Terminate Event

Reopening the Document in Read Mode

Object	*Formula or Event*
Form	Initialize Event
Form	Window Title Formula
Form	Query Open Event
Subject Field	Initialize Event
From Field	Initialize Event
Counter Field	Initialize Event
DisplayNum Field	Value Formula
DisplayNum Field	Initialize Event
Body Field	Initialize Event
Form	PostOpen Event

Toggling from Read Mode to Edit Mode with Document Open

Object	*Formula or Event*
Form	QueryModeChange Event
Subject Field	Entering Event (depends on cursor)
Form	PostModeChange Event

Toggling from Edit Mode to Read Mode with Document Open (No Changes)

Object	*Formula or Event*
Form	QueryModeChange Event
Form	PostModeChange Event

Toggling from Edit Mode to Read Mode with Document Open (Saving Changes)

Object	*Formula or Event*
Form	QueryModeChange Event
Same sequence as for saving a document	
Form	PostModeChange Event
Form	QueryClose Event
Same sequence as for closing a document	
Same sequence as for reopening a document in read mode	

Moving Cursor From One Editable Field to Another

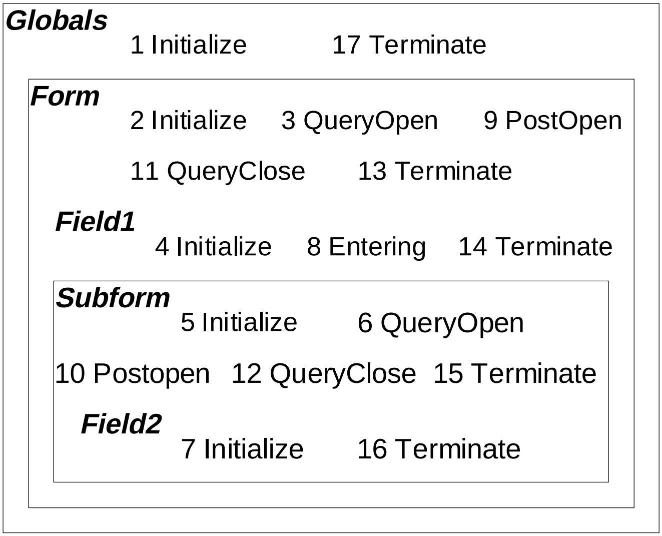
Object	*Formula or Event*
First field	Exiting Event
Second field	Entering Event

Refreshing Fields While in Edit Mode (F9)

Object	*Formula or Event*
Subject Field	Input Translation formula
Counter Field	Value Formula
DisplayNum Field	Value Formula
Subject Field	Input Validation Formula
Form	PostRecalc Event

Sequence of Events in a Complex Example

The following picture shows the sequence of events in a form which contains a subform.



Here is the sequence of events when the document is opened:

1. Initialize of Globals
2. Initialize of Form
3. Queryopen of Form

4. Initialize of Field1 (contained in Form)
5. Initialize of Subform
6. Queryopen of Subform
7. Initialize of Field2 (contained in Subform)
8. Entering of Field1
9. Postopen of Form
10. Postopen of Subform

This is the list of events when the document is closed:

11. Queryclose Form
12. Queryclose Subform
13. Terminate Form
14. Terminate Field1
15. Terminate Subform
16. Terminate Field2
17. Terminate Globals

LotusScript Programming Tips and Considerations

The following section will give you some help structuring your LotusScript code for event programming within Lotus Notes.

General Suggestions

Do any or all of the following to improve your scripts:

- Declare all variables in the global definitions for an object and use the *Option Public* statement. Then instantiate the variables in the PostOpen event or in a subroutine that you can call from either the QueryOpen event (for an existing document) or the PostOpen event (for a new document). Your variables will be easier to find and maintain, and you'll be able to use them in any script for the object. Also, you might consider using Option Declare to make certain that you have declared all the variables in your application.

  For an example of declaring variables this way, see the global definitions area in the Memo form in the Mail (Release 4) template (mail4.ntf).

- Store subroutines and functions in the global definitions for a form or navigator. Then you can use the subroutines or functions with any object on the form or navigator. For an example, see the script for the SaveDialog event in the global definitions for the Memo form in the Mail (Release 4) template (mail4.ntf).
- If the script for a single event approaches the limit of 64K, the script is probably much too long for easy maintenance or understanding. Use functions and subroutines to split the script up.
- To re-use a segment of script in multiple scripts, put the segment into a function or subroutine, or use script libraries (see section on script libraries further on in this chapter).
- Try not to nest subroutine calls or conditionals deeper than three levels. Nesting to too many levels makes scripts hard to follow.
- To debug a script that runs on a shared field, insert the field into a temporary form so that you'll have a place from which to run the debugger.
- In Initialize and Terminate events in forms, fields, actions, and buttons, avoid using the uidoc variable for the NotesUIDocument class. A document object may not be available to access (for example, a document window may not be open) at the time the script runs.

For complete information on LotusScript, see the online Lotus Notes help information or the *Programmer's Guide.*

Use Consistent Variable Names

The Notes templates use a set of standard variable names as shown in the table below. For example, in the Notes templates the variable *note* always refers to the current back-end document.

Using these names in your own scripts makes your scripts easy to read and understand, keeps them consistent, helps you maintain them more easily, and may help you share them with other developers.

Consider using all lowercase for object variables and a combination of lowercase and uppercase, for example VariableName, for other variables.

When passing values to a subroutine or function, use the same variable names in the called routine as in the calling routine. For example, don't call something StatusNumber in one and StatNo in the other. Consistent naming ensures that others can easily read and understand the script.

Class Name	*Object Variable*	*Comments*
NotesSession	session	
NotesDatabase	db	
NotesView	view	
NotesViewColumn	column	
NotesDocument	note	Refers to the data associated with the current document
	parent	The parent of the current document
	child	A child of the current document
	profile	A profile document from which you are retrieving processing parameters
NotesItem	item	
NotesRichTextItem	rtitem	
NotesEmbeddedObject	embobj	
NotesDocumentCollection	documents	
	responses	Use if you are working within one collection of responses to the current document
	children	An alternative to using the variable name responses. Use if you're using child as the NotesDocument object variable.
NotesDateTime	date1, date2, ...	Consider using for comparing dates
NotesAcl	acl	
NotesAclEntry	aclentry	
NotesAgent	agent	
NotesDbDirectory	dbdir	
NotesLog	log	
NotesUiWorkSpace	ws	
NotesUiDocument	source	Already an argument to the form events - using this name keeps your scripts consistent
	uidoc	To use, set uidoc = source in PostOpen. Then you can use this object variable in field and action scripts in the form.

Using Script Libraries

A script library is a place where you can store code segments you want to use from other scriptable objects. You may code options, declarations, an initialize subroutine, a terminate subroutine, and user scripts.

To write a new script, enter a statement such as Function or Sub in an existing script. The editor automatically creates a new script and transfers your code there.

To incorporate a script library into a scriptable object, enter a *Use* statement in the (Options) script for the object or for the (Globals) object. For example, to make the script library named SCRLIB1 available to a form's scripts, enter the following statement in the (Declarations) script for the form:

```
Use "SCRLIB1"
```

The name is case-insensitive and should not contain spaces. Specify the name as a character literal or named constant:

```
Const lib = "SCRLIB1"
Use lib
```

The code in the (Options), (Declarations), Initialize, and Terminate scripts of the library becomes available as though it were in the current object's corresponding scripts. User scripts in the library become available as though they were in the current object.

Catching Errors at Compile Time

Specifying *Option Declare* at the beginning of your LotusScript module forces you to declare variables explicitly. With this option in effect any undeclared variables will be flagged during compile time. This is useful if you design large applications and it prevents you from searching for typing errors.

Improving Form Performance

A form that performs well is one that Notes can calculate quickly for display, so that documents created with the form are more likely to open quickly.

To improve form performance, do any or all of the following:

- Avoid over-using hide-when formulas on forms. Each formula that Notes must calculate when opening a form slows your application down. Before you use a hide-when formula, try using a computed subform or a hide-when condition such as "ide when editing"or "ide when reading."

- If you must use hide-when formulas to hide buttons on an action bar, use @Command([RefreshHideFormulas]) or the LotusScript RefreshHideFormulas method in the action formulas or scripts to force calculation of the hide-when formulas. This closely correlates the appearance of different buttons with users' button clicks, and allows each calculation to occur only when needed.
- If a form has keywords fields — for example, in a layout region — and you want formulas to calculate based on changes in those fields — for example, hide-when formulas that progressively disclose items in the layout region — enable the "Refresh fields on keyword change" option instead of the "Auto refresh fields" option. Notes performs more calculations when "Auto refresh fields" is enabled — for example, it refreshes all formulas every time a user moves between keyword fields, instead of just when values in keyword fields change.

  Note Examine applications that were originally created in Release 3.x and make this change where appropriate.
- Remove infrequently used items from a form. For example, redesign your application to display infrequently used items in a dialog box.
- Consider limiting or eliminating entirely the use of shared fields or subforms on any form that must open quickly.
- Minimize the number of fields per form, because each field is calculated when a document is opened, refreshed, or saved. After your design is complete, run an agent to remove any blank, unused fields.
- Consider putting field formulas into form events rather than into the fields themselves, so you can more easily control which formulas are calculated at each event. Don't use hidden fields for processing events.

If your application was created in Release 3.x, it may include forms with hidden fields containing formulas that process a document when it's opened or saved. To improve the performance of the application, convert the formulas to LotusScript, and use the PostOpen and QuerySave form events.

When to Use Formulas and LotusScript

In general, formulas are best used for working within the object that the user is currently processing, for example, to return a default value to a field or to determine selection criteria for a view. Scripts are best used for accessing existing objects, for example, to change a value in one document based on values in other documents. Scripts provide some capabilities that formulas do not, such as the ability to manipulate a database Access Control List (ACL). Formulas provide better performance in some situations and may be more convenient for simple applications.

When you're ready to use both, deciding whether to use LotusScript or the Notes formula language for a given task usually depends on the complexity of the task. Consider these questions when making your decision:

- **Do you need to process a quantity of data?**

  A formula that "touches" many databases or documents using @Functions must rely on the Notes user interface to access each document, whereas LotusScript accesses the documents more efficiently and quickly.

  For example, LotusScript is a good tool for creating an agent that scans all the databases on your workspace and returns information such as size of database, percent used, number of documents, and so on. LotusScript is also a good tool for running a full-text search on multiple documents and performing an action with the results of the search.

- **Are you using front-end or back-end LotusScript Notes classes?**

  The LotusScript user interface (front-end) classes use the same Notes code as their equivalent @Commands, so LotusScript won't perform better than the formula language when you use these classes. The database (back-end) classes, however, use different code, and perform more quickly than the equivalent @Functions.

  For example, avoid using the front-end class NotesUIDocument to do many field updates. The back-end class NotesDocument is much faster, and allows you to assign data types (including rich text) and to add new (hidden) fields. The front-end class allows you to update only fields that already exist on the form, and it allows you to insert only text in the field, as @Command([EditInsertText]) does.

  In addition, the front-end classes will not work in scheduled agents run by a server, only in agents run from a user's workstation (for example, from the menu).

- **Do you need to manipulate the currently selected object?**

  Use the formula language instead of LotusScript.

- **Do you need to program buttons on an action bar?**

  Consider using the formula language instead of LotusScript. Button actions are usually simple and perform tasks usually accomplished directly through the Notes user interface such as saving or closing a document.

- **Do you need to return the default value to a field?**

  Use the formula language instead of LotusScript.

- **Do you need to return the title of a window?**

  Use the formula language instead of LotusScript.

- **Do you need to control a workflow process from a form?**

  LotusScript is best for controlling workflow with form events, especially the QuerySave event, because it can handle the more complex tasks you may want to accomplish, such as looping, and setting multiple variables.

  For example, you can require a user to fill out fields on a form in a pre-determined order by manipulating enter and exit field events, or you may prevent a user from opening, saving, or editing a form until certain conditions are met.

- **Are you including too many @Functions in one formula?**

  If a formula includes many @Functions in sequence, try changing the formula to LotusScript. However, formulas that need only a single @Function, such as @Command[FilePrint], are more efficient and perform better than scripts that do the same thing.

Using the Evaluate Function to Combine LotusScript and Formulas

Use the Evaluate function in LotusScript to combine pieces of formula language with LotusScript. This allows you to make your scripts leaner wherever @Functions do something in fewer lines than LotusScript does. Keep in mind that including formulas in scripts may make the scripts easier to write, but won’t necessarily improve performance.

You can use Evaluate to include any @Functions except the ones that directly interact with the Notes user interface, such as @Prompt, @DialogBox, @PickList, and @Command. A couple of particularly useful @Functions to combine with LotusScript are:

- @Name, which lets you manipulate hierarchical names
- @Replace, which pulls a value from a text list without requiring the looping that LotusScript would demand
- @Unique, which removes duplicates from a text list

You can also combine LotusScript and formulas in an application by using them in different parts of the same form. For an example, see the DocumentWorkflow subform in the Document Library (Release 4) template (doclib4.ntf). The actions that trigger the workflow are written using @Functions, forcing the user to save the document. But the actual workflow and dataflow logic is in LotusScript, initiated in the QuerySave event.

Evaluate Function in LotusScript

The Evaluate function executes a LotusScript formula.

Syntax:

`Evaluate(`*`macro`*` [, `*`object`*`])`

Elements:

macro

Mandatory. The text of the Notes macro, in the syntax that Notes recognizes. Refer to the Notes documentation for the correct syntax of the macro.

Note The macro text must be known at compile time, so use a constant or string literal. Do not use a string variable.

object

Optional. (If the macro requires a Notes object)

Example:

```
Evaluate("@Sum(Numlist)", ...)
or
Const NotesMacro$ = "@Sum(NumList)"
Evaluate(NotesMacro$, ...)
```

The next example is incorrect because a string variable is used.

```
NotesMacro$ = "@Sum(NumList)"
Evaluate(NotesMacro$, ...)
```

Return value

If the macro being run returns a value, the Evaluate function returns a Variant containing that value. Otherwise, the function does not return a value.

Sample code

This script runs when the user exits from the Subject field and changes the characters to propercase.

```
Sub Exiting (Source As Field)
   Dim ws    As New NotesUIWorkspace
   Dim uidoc As NotesUIDocument
   Dim doc   As NotesDocument
   Dim eval  As Variant
   Set uidoc = ws.CurrentDocument
   Set doc = uidoc.Document
   eval = Evaluate( "@ProperCase(Subject))", doc)
   Call doc.ReplaceItemValue("Subject", eval)
End Sub
```

In this example we use the *Evaluate* function to get *@ProperCase* carried out. Parameters to the Evaluate function are the string containing the @Function and the field name as well as the object that contains the field.

Tip Since in the above example the variable *uidoc* is only used to get the object of the next lower class you may also write *Set doc=ws.CurrentDocument.Document* to initialize variable *doc*.

Making Field Value Changes Effective

There are two ways of making changes to field values in your LotusScript programs effective.

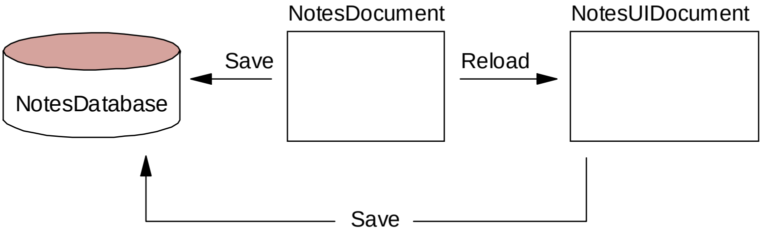
You can use the Refresh method of the NotesDocument class. It has the same effect as using the Refresh key on the Lotus Notes user interface.

When you modify ("ReplaceItemValue" or remove) "RemoveItem" fields in a document in your LotusScript program, you need to use the Reload method of the NotesDocument class to make the changes effective in the Lotus Notes user interface. The following statements are examples to show the Reload method.

```
Postopen(Source As Notesuidocument)
  note.RemoveItem("Action")
  note.RemoveItem("SaveOptions")
  note.ReplaceItemValue("Action","Approve")
  source.Reload
  .....
```

Note You usually describe the following statement at the initialization stage of your program to improve performance. Remember to perform the Reload method in your programs when you use this statement.

```
source.AutoReload = False
```



Using Validation Formulas and QuerySave

If you are using Input Translation and Input Validation formulas along with QuerySave, be sure to do a refresh (source.Refresh) at the beginning of the script for the QuerySave event. The reason for doing this is that the QuerySave event occurs before Notes refreshes the document when saving.

You want QuerySave to have the properly validated data to process. For example, you don't want QuerySave to process an empty field because a validation formula that would have flagged the field as empty hasn't yet run.

Working With a Rich Text Item

The NotesRichTextItem class inherits from NotesItem class. Therefore, you may use GetFirstItem of the NotesDocument class to access the rich text item. Of course, you also may use methods of NotesRichTextItem, for example GetFormattedText. However, when a rich text item is the return value of a method such as GetFirstItem in NotesDocument, do not declare it with a Dim statement, but leave it a variant. It cannot be declared as a NotesRichTextItem object — a "type mismatch" error occurs. If you declare it as a NotesItem object, you cannot use the NotesRichTextItem property and methods — a "not a member" error occurs.

This is a piece of code to check if an item is a rich text item:

```
REM assume variable doc set
Set item = doc.GetFirstItem("Field1")
If item.Type <> RICHTEXT Then
  MessageBox "Field1 is not a rich text item"
End If
```

This button example formats a rich text item as a text string.

```
Sub Click(Source As Button)
     Dim ws    As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Dim doc   As NotesDocument
     Dim item  As Variant
     REM Do not Dim item either as NotesItem
     REM or NotesRichTextItem
     Set uidoc = ws.CurrentDocument
     Set doc = uidoc.Document
     Set item = doc.GetFirstItem("Body")
     Messagebox item.GetFormattedText(False, 40)
End Sub
```

Note If you are creating a new document, you must save it before you can access a rich text field.

The shortcoming of rich text fields is that updates to such fields are not visible to the user immediately. In order to see the rich text that has been added, the user must first save the document, quit out of the document and then reopen it. The following sample script shows one way to automate this process.

```
Dim ws As New NotesUIWorkspace
Dim s As New NotesSession
Dim db As NotesDatabase
Dim uidoc As NotesUIdocument
Dim view As NotesView
Dim doc As NotesDocument
Dim searchdoc As String
Dim rtitem As Variant

Set db = s.currentdatabase
Set uidoc = ws.currentdocument
Set doc = uidoc.Document

'** Create New RichTextItem in the current document
Set rtitem = New NotesRichTextItem(doc,"RT")

'** Create Attachment
Call rtitem.EmbedObject ( EMBED_ATTACHMENT, "", _
       "c:\embed.txt", "Att" )

'** Set the form field
'** Save document through back-end
doc.Form = "Add Rich Text through UI"
Call doc.save(False,False)

'** create unique key
searchdoc = doc.UniversalId

'** Refresh the main view through back-end and
'** front-end methods
Set view = db.getview("Main")
Call view.Refresh
Call ws.viewrefresh

'** Set save options to zero so that user does not
'** get prompted to save after closing uidoc
doc.saveoptions = "0"
Call uidoc.close

'** Open database to main view
'** Find the document again based on searchdoc
'** Open document
Call ws.OpenDatabase("","","Main",searchdoc,False,True)
Set uidoc = ws.Editdocument(True)

'** Reset doc and remove fields
Set doc = uidoc.document
Call doc.RemoveItem("SaveOption")
```

The script works as follows:

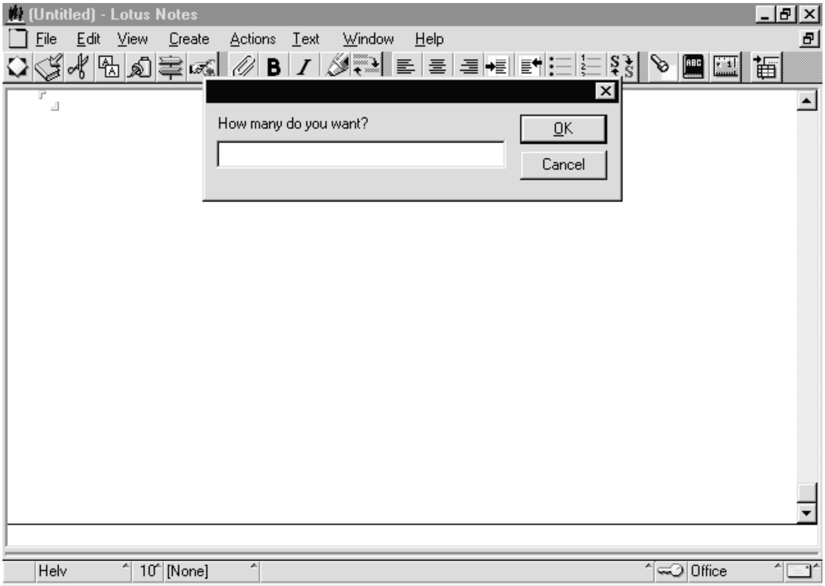
1. Creates a new Rich Text field in the current document.
2. Attaches a file to this Rich Text field.
3. Sets the appropriate form field and saves the document.
4. Retrieves the unique document ID.
5. Gets a handle to the view called “main” and refreshes the view’s index through front-end and back-end methods, so the new document will display in the view.
6. Sets SaveOption to “ ” so the user will not be asked to save the document, and then closes the document.
7. Using the OpenDatabase method, opens the “main” view (which is sorted by unique document ID) and selects the appropriate document.
8. Reopens the document using the EditDocument method, and removes the SaveOptions field. (Instead of removing SaveOptions it may also be set to “ ”)

Prompting for User Input

Inputbox Statement

```
Dim num As Integer
num% = CInt(InputBox$("How many do you want?"))
```

The following picture shows the result of the above statements.



How many do you want?

Note If you need multiple input items, it is recommended to use the Dialogbox statement rather than the Inputbox statement to avoid a large number of input boxes for the users.

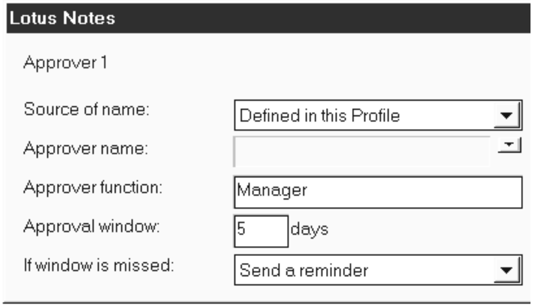
Dialogbox Statement

Some Dialogbox statements are used in the Approval Cycle template. This statement needs a prepared form with one layout region for a dialog box.

(Declarations) in ApprovalLogic Subform

```
   Dim ws As NotesUIWorkspace
 Postopen(Source As Notesuidocument) in ApprovalLogic Subform
Set ws = New NotesUIWorkspace
 GetApproverDetails in ApprovalLogic Subform
   If ws.DialogBox("(ApproverInfo)", True, True) = False Then
     Exit Sub
   End If
```

The following dialog box is displayed by the above statements as implemented in the Approval Database.



Lotus Notes

Approver 1

Source of name: Defined in this Profile

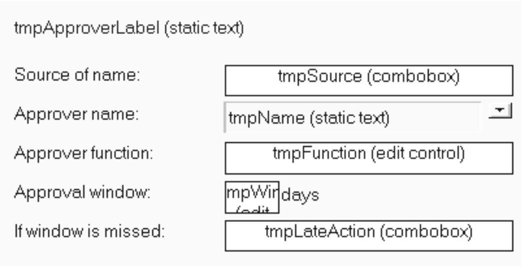
Approver name:

Approver function: Manager

Approval window: 5 days

If window is missed: Send a reminder

(ApproverInfo), which is the first argument of the Dialogbox statement, refers to the following layout region of the (ApproverInfo) form.



tmpApproverLabel (static text)

Source of name: tmpSource (combobox)

Approver name: tmpName (static text)

Approver function: tmpFunction (edit control)

Approval window: mpWir (edit days

If window is missed: tmpLateAction (combobox)

Note Once a dialog box is called, all fields defined in the dialog box are created in the original document from which the dialog box is activated. If the same fields in the dialog box exist in the original document, values inputted in the dialog box are copied to corresponding fields in the original document.

Guidelines for Presenting Dialog Boxes Using Formulas Versus Scripts

The following guidelines may help you decide which way of presenting a dialog box is most appropriate in a particular area of your application.

How	*Why*
@Prompt formula	You want to present a simple message, or a keyword list for selection, at a place in Notes where it's appropriate to use a formula, such as an action. You want to present a message without returning the user's input as a value, or present a message that returns simple input based on which buttons the user clicks. OK / Cancel buttons or Yes / No / Cancel buttons are adequate button combinations.
@DialogBox formula	You want to use a custom dialog box to present fields modally, so that the user must complete the fields before proceeding with other work. If you don't want to control the user's progress in this way, present the form itself without a dialog box. You want to use a form, perhaps with a layout region, to present the fields attractively. You may also want to use progressive disclosure to reveal fields and other objects in a certain order.
@PickList formula	You want to present lists to the user and perform actions based on the user's selections from lists. These lists may come from address books or views in databases.
DialogBox method in LotusScript	You have the same goals as the designer who uses @DialogBox, but you prefer to work in LotusScript. Alternatively, the place in Notes where the dialog box will appear — for example, a form QuerySave or PostOpen event — is more appropriately coded with LotusScript.
MessageBox function (returns a value) or statement (does not return a value) in LotusScript	You want to present some information to the user or ask a Yes / No question, and the place in Notes where the dialog box will appear — for example, a form QuerySave or PostOpen event — is appropriately coded with LotusScript. You may also want to use some combination of buttons other than OK / Cancel or Yes / No / Cancel. Settings for MessageBox are available in LSCONST.LSS which is installed in the Notes data subdirectory.

Error Handling

Ideally you would not need to write anything to handle run-time errors, however, some errors may happen at run time, such as running out of disk space or dividing by zero. A script will stop unexpectedly when such run-time errors happen. To avoid this situation, you can write error-handling procedures in your script.

Using On Error and Resume Statements

Using On Error and Resume statements in your script, you can handle run-time errors that may occur. These statements are built-in functions provided by LotusScript. The script needs the following steps to handle the error.

1. First, trap the error using an On Error statement and specify where to go to handle the error.

   For example, if the error happens, you can go to the label ERRORPROC.

```
     Dim x As Integer, y As Integer, z As Integer
     x = 3
     y = 0
     On Error GoTo ERRORPROC
     z = x/y
Exit Sub
ERRORPROC:
```

2. Second, script the error-handling process. For example, at the ERRORPROC: label.

```
ERRORPROC:
    MessageBox("Divide error")
    y = CInt( InputBox("Enter new number") )
```

3. Third, complete the error-handling process using a Resume statement to go back to the statement where the error occurred.

```
     Dim x As Integer, y As Integer, z As Integer
     x = 3
     y = 0
     On Error GoTo ERRORPROC
     z = x/y
Exit Sub
ERRORPROC:
     MessageBox("Divide error")
     y = CInt( InputBox("Enter new number") )
     Resume
```

Creating an Error Handler for Debugging

It is useful to have an error handler to help debug your programs, as the LotusScript debugger ends when errors occur. To prevent this from happening, you can create an error handler like this:

```
On Error Goto ErrorHandler
ErrorHandler:
  Messagebox "Error:" & Error(Err), 0+64, "Error!!"
  Print "Error No. : " Err
  Print "Description : " Error(Err)
  Print "Line No. : " Erl
  Resume Next
  Exit Sub
```

Note If you include the constant definition file (%Include "LSCONST.LSS", you can use constant symbols (MB_OK, MB_ICONINFORMATION and so on) instead of values 0 and 64 in the Messagebox statement.

Note If you want to catch all errors in your programs, you need to write the above error handler in all event routines you described (for example, "initialize" "postopen" and so on), but not in the subroutines. Once you have written an error handler in a specific event routine, it can be referred to in the subsequent subroutines.

Using the Debugger

While you are writing scripts, you will find some errors which will require fixing. Notes recognizes two kinds of LotusScript errors: Compile errors and run-time errors.

Compilation of your script takes place when you save it. Compile errors are reported and the script cannot be saved. Because Notes does not allow you to save your script with compile errors, you have to correct all the compile errors first.

Tip If there are compile errors in your LotusScript programs, you cannot usually save your design. If you want to do so, you can exclude the statements with compile errors using the %REM and %ENDREM statements as shown in the following.

```
%REM
"Your program with errors"
%ENDREM
```

A run-time error is an error which cannot be detected during compilation. Run-time errors are found while Notes is running the script. If your script includes run-time errors, it will stop when the error happens, and you may not understand the reason why the script stopped. The script may have the correct syntax, but, for example, division by zero is not allowed. There is a simple example of this run-time error:

```
Dim x As Integer
Dim y As Integer
Dim z As Integer
x = 5
y = 0
z = x / y
```

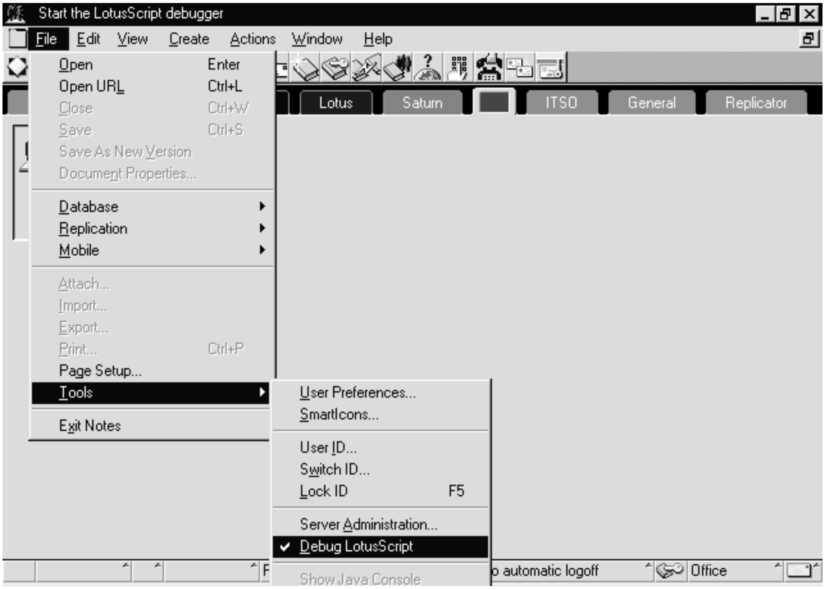
During execution of the above code LotusScript will stop and issue an error message because dividing 5 by 0 is not a valid operation.

There is one more error type, which is a logical error. You might be able to run your script without errors, but the result is not as intended.
The Debugger helps you to detect run-time errors and logical errors.

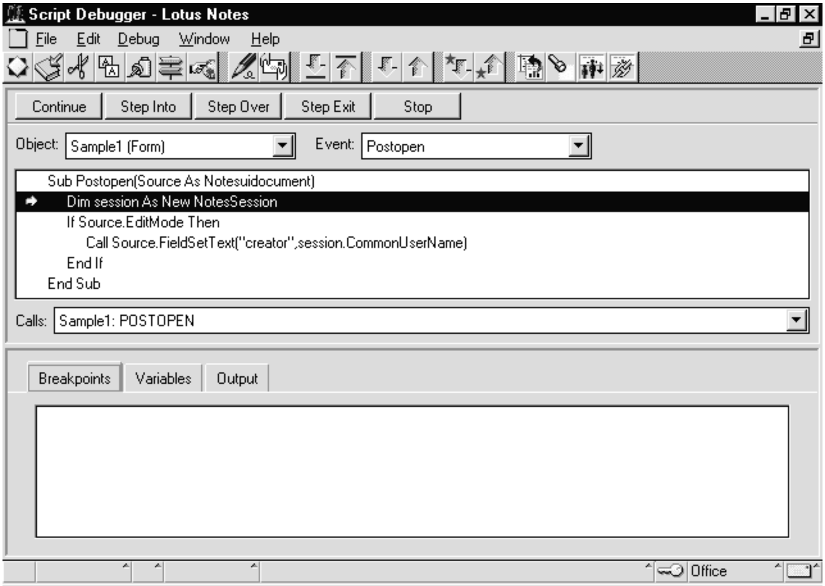
How to Enable the Debugger

It is easy to enable debug mode. Before running your script:

1. Choose File - Tools - Debug LotusScript.
2. To check if the Debugger is enabled, choose File - Tools. If the Debugger is on, there is a checkmark next to the menu option, as shown in the following figure. If you click on the Debug LotusScript menu again, debug mode is disabled.



If the Debugger is enabled, when you start running any LotusScript the debugger is launched and the script stops at the first line. The debugger is shown as follows:



Object: Sample1 (Form)

Event: Postopen

```
Sub Postopen(Source As Notesuidocument)
    Dim session As New NotesSession
    If Source.EditMode Then
        Call Source.FieldSetText("creator",session.CommonUserName)
    End If
End Sub
```

Calls: Sample1: POSTOPEN

In this example, the script is in interrupt mode.

When you run a script in debug mode, the script shows one of three states:

- When a script is interrupted at a breakpoint, the debugger has control.
- When a script is stepping, control passes to the script and then back to the debugger after a single statement in the script is performed.
- When a script is continuing, it runs uninterrupted until a breakpoint is reached.

While the script is in interrupt mode, you can do one of the following:

- Inspect the script
- Inspect the value of variables and properties
- Control which is the next statement that will be performed
- Inspect other defined objects, events and the scripts related to them

You can control which is the next statement that will be performed in interrupt mode. Click on:

- Continue

  To Continue until a break point is reached

- Step Into

  To perform the current statement and step to the next statement

- Step Over

  To perform the current statement and step to the next statement, stepping over the subprogram if the current statement calls a subprogram

- Step Exit

  To continue executing the current subprogram and stop in the subprogram that called it at the line following the call

Making Breakpoints

If you find an error that is a run-time error or a logical one, inspect your script and make breakpoints at the statement (or around it) where you suspect the error is occurring. Then run your script. It will stop at the breakpoint. In interrupt mode, you can inspect the value of important variables and properties.

One-Step Execution

You can perform only the current statement. Then you can inspect the difference of some important values of variables or properties between, before and after performing the statement.

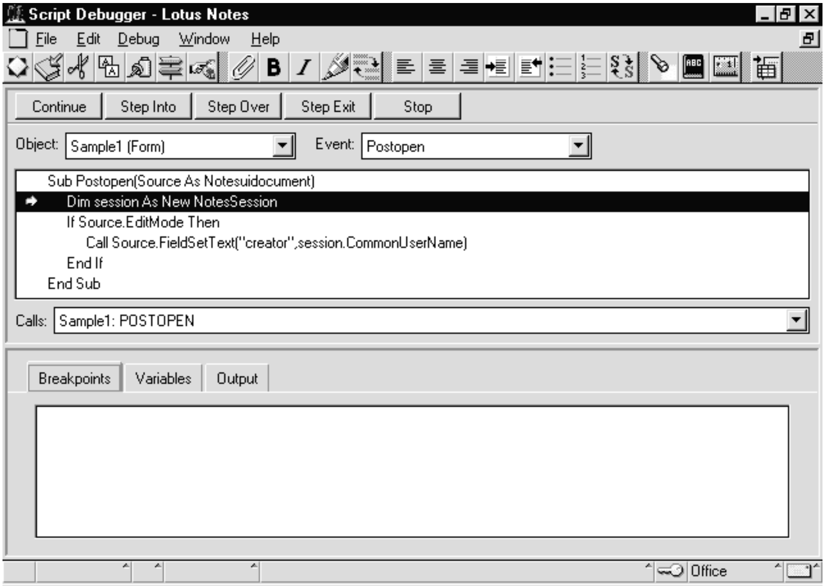
Instance Inspection

1. Click the Variables tab in the bottom pane. Or, choose Debug - Variables in debug mode to access the variables window. The variables defined for the procedure appear in a three-column display, showing the name, data type, and value of each variable.
2. To view array or type members, click the arrow to the left of the variable name.

A Simple Example

To show you how to use the debugger, we will take the database we used as an example for the *PostOpen* event.

1. Choose File - Tools - Debug LotusScript to enable debug mode.
2. Open the database and choose Create - Sample1. The following figure shows that the debugger has been launched. The script has stopped.



Object: Sample1 (Form)

Event: Postopen

```
Sub Postopen(Source As Notesuidocument)
    Dim session As New NotesSession
    If Source.EditMode Then
        Call Source.FieldSetText("creator",session.CommonUserName)
    End If
End Sub
```

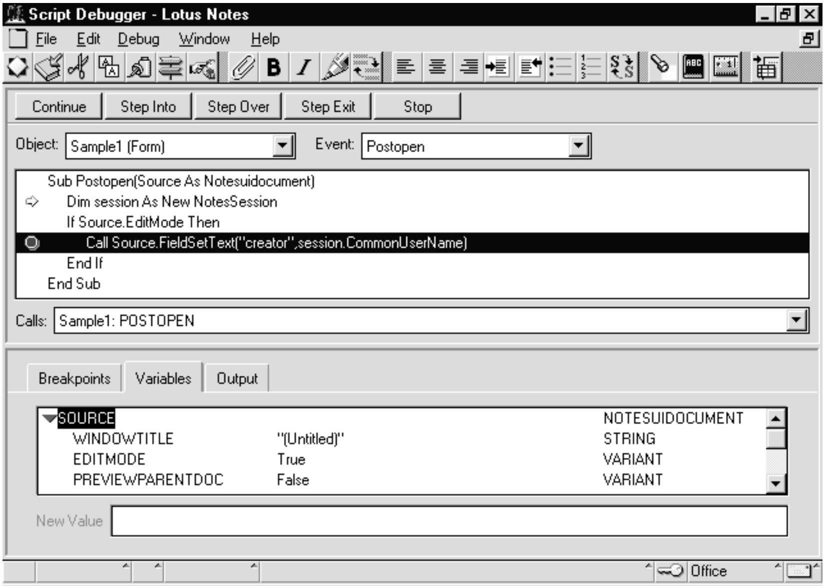
Calls: Sample1: POSTOPEN

The script added to the Sample1(Form) object has been launched by the Postopen event and the execution stops at the marked sentence.

Go through the debugger.

3. Double-click the statement *Call Source.FieldSetText....* in the upper pane. This creates a breakpoint.
4. Click the Variables tab. In this pane you can see instances or variables.

5. Click the green triangle next to Source in the bottom pane. You can see properties of the Source instance which is of type NotesUIDocument. This class represents the document that is currently open in the Notes workspace.



Object: Sample1 (Form)   Event: Postopen

```
Sub Postopen(Source As Notesuidocument)
   Dim session As New NotesSession
   If Source.EditMode Then
      Call Source.FieldSetText("creator",session.CommonUserName)
   End If
End Sub
```

Calls: Sample1: POSTOPEN

SOURCE		NOTESUIDOCUMENT
WINDOWTITLE	"(Untitled)"	STRING
EDITMODE	True	VARIANT
PREVIEWPARENTDOC	False	VARIANT

New Value

6. You can see the variable *session* does not yet have values.

7. Click the Continue action button. Or, choose Debug - Continue.
The script runs and stops at the breakpoint that you made. You can see that the *session* variable now has a value.

8. Click the Continue action button. The debugger closes.

This very simple example shows how easy it is to control the execution flow of the program, and to inspect variables.

Tracing Your Programs Without a Debugger

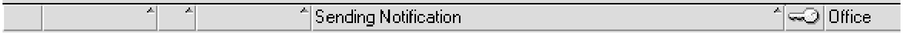
There are some ways to trace your programs without a debugger, though you need to add some statements in your programs for them. You can use the PRINT and MESSAGEBOX statements to look at variables in your programs.

Print Statement

The Print statement displays constant values and the contents of variables on the status line at the bottom of the Notes interface.

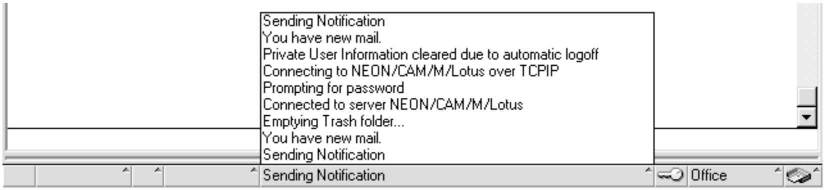
```
Print "SendingNotification"
```

This statement results in the following:



Sending Notification

When you click the status line with the mouse, you can see the following message list box, which contains the Print message history.

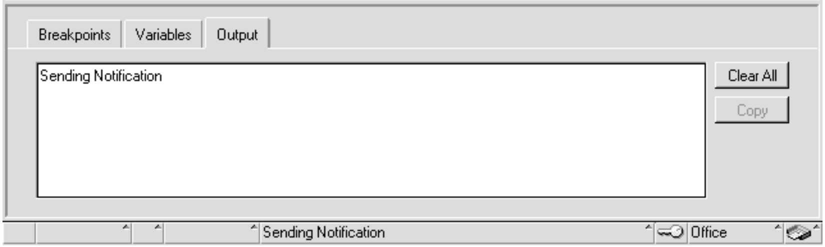


```
Sending Notification
You have new mail.
Private User Information cleared due to automatic logoff
Connecting to NEON/CAM/M/Lotus over TCPIP
Prompting for password
Connected to server NEON/CAM/M/Lotus
Emptying Trash folder...
You have new mail.
Sending Notification
```

Sending Notification

To clear the status line simply issue the Print statement with no arguments. This clears the status line. However, you can still click on the cleared area to display the message box.

You can also see the messages created by the Print statement by clicking the Output button when using the Lotus Notes debugger:



Sending Notification

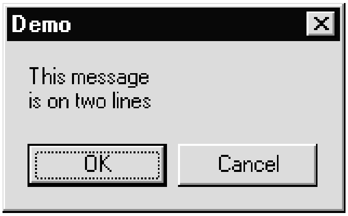
Sending Notification

Messagebox Statement

The Messagebox statement displays a dialog box with some buttons to show messages.

```
%INCLUDE "LSCONST.LSS"
Dim twoLiner As String
twoLiner = |This message
is on two lines|
MessageBox twoLiner, MB_OKCANCEL, "Demo"
```

The following message box is displayed.



Demo

This message
is on two lines

Note The vertical bar (|) is the string delimiter for multi-line strings.

External Tools

The Notes API

The Notes API lets you write a program that processes data in Notes, or moves data in and out of Notes. The API accesses the Notes database layer, much as the Notes user interface itself accesses it. You can also use the API to access the server software, the Tools menu in the workstation software, and the File Types list in the File Export dialog box.

You can write an API program to:

- Extract external data, reformat it, and store it in Notes.

  For example, you can retrieve information from SQL records.

- Extract Notes data, reformat it, and store it in an external application.

  For example, you can retrieve Notes workflow status data into a word processor or executive information management (EIS) system.

- Add commands to the File - Tools menu.

  For example, when a user chooses your new command, Notes can launch your program and pass user context information to it, such as which view is active, whether the user is editing a document, and which field contains the cursor. Your program can compute new values and enter them into Notes fields.

- Implement server add-in tasks.

  For example, you can implement a task that takes conditional actions beyond Notes background macro capabilities. A server add-in task functions as a daemon. It has no user interface and runs in the background like other server tasks.

- Create a custom file export format.

  For example, when a user selects your new file type in the Notes File Export dialog box, Notes launches your program and exports Notes data to it.

  For more information about the Notes API, see the Notes API *User's Guide.*

Summary

When developing applications for Lotus Notes, you can choose the appropriate tool for the job based on:

- Skill level
- Application requirements
- Operating platform

You choose the programming language and APIs:

- Notes Simple Actions
- Notes Formulas
- LotusScript
- Lotus Notes C API
- Lotus Notes C++ API
- Interflox for OS/2 REXX

Chapter 7
Using the LotusScript Extensions Toolkit

This chapter introduces a powerful technique to enable your Notes applications to access and control external applications. It is based on the open object model of the LotusScript language which you can extend by adding new classes.

The first section provides you with an overview of LotusScript extensions. Then, we will introduce a toolkit to you that allows you to develop your own extensions.

You will be particularly interested in this chapter when you are highly experienced with C++ and LotusScript.

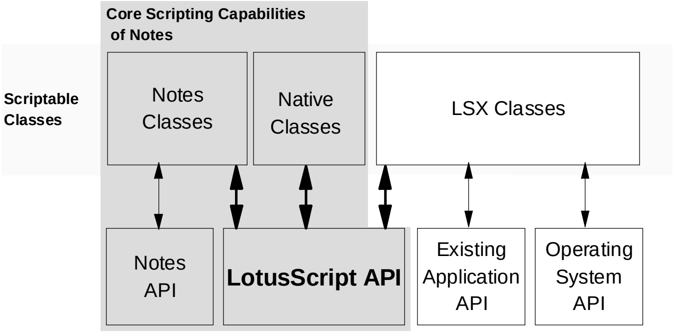
What Is an LSX?

A LotusScript Extension (LSX) is a dynamic library of LotusScript classes written in the C++ programming language. You can use these classes just like any other LotusScript class in your event scripts. For example, you can create new objects from those classes, invoke methods, and get their properties.

LSXs provide you with a third kind of LotusScript class. The first two are intrinsic to Lotus Notes: classes defined within LotusScript itself using the Class statement, and classes that represent Notes objects, for example Notes databases and documents.

The source programming language of LSXs is C++, which enables you to use APIs of some other application. After an LSX is loaded by Notes, the LSX registers its C++ class definitions as corresponding LotusScript classes. This means an LSX extends the functionality of LotusScript running in Notes, because it enables any Notes application to connect to resources and functionality of external applications.

The following figure shows the extended LotusScript capabilities introduced by LSXs:



Using an LSX

Several LSXs for access to Relational Database Management System as well as transaction system integration are available. If none of the existing LSXs fits your needs, you have the option to develop your own LSX. You are supplied with a LotusScript Extension Toolkit that facilitates the mapping of your C++ code to LotusScript classes.

You can use all the classes of an LSX in your LotusScript event scripts by putting the USELSX statement in the “Options” event of the “Globals” definition section of a script.

Note The USELSX statement offers you two options. You can pass the LSX name as a library filename including the full path (USELSX C:\MYLSX\SAMPLE.DLL), or you can use the name that is associated with that LSX in the LSX class registry. For example, if the LSX is registered as MYSAMPLE=C:\MYLSX\SAMPLE.DLL, the statement looks like USELSX *MYSAMPLE. You may prefer the latter because it is much more location independent.

As soon as you leave the “Globals” section, Notes loads and registers the LSX. All new classes are now available for your event scripts. LotusScript performs a type-check of all references to the new classes in your script against the registered class definitions. Furthermore, when you select Show browser in the programming pane to view the browser, and select Notes:classes in the combo box, the registered LSX classes are displayed, including their properties and methods.

If you declare an LSX twice, using the USELSX statement, LotusScript will use the LSX library that is already loaded.

Finally, it should be noted that an LSX is not so tightly coupled with Notes. It only interacts with the LotusScript interpreter embedded in Notes to provide the functionality. For that reason, you may use an LSX in any Lotus product that supports a LotusScript interpreter of Release 3 or higher.

Using the LSX Toolkit

This section describes the LSX Toolkit. You will understand its architecture and how to develop a new LSX.

Overview

The LSX Toolkit is a software development environment that enables you to implement new LSXs in the C++ programming language.

When Do You Need the LSX Toolkit?

There are certain situations that may lead you to develop your own LSXs:

- Need to access external applications.

  For example, an LSX may define classes to access a specific DBMS, a document management system, or even execute some FORTRAN code.

- Need to access part of an operating system.

  For example, an LSX may define specialized classes to gain access to system resources such as the window system or communication facilities.

- Need to implement an algorithm where efficiency or code-size requirements make it undesirable to implement in LotusScript.

  In this case, the LSX is not being used to script an extra application, but as an alternative to using LotusScript itself. The LSX classes are an alternative (or a supplement) to LotusScript native classes.

- In cases where you wish to preserve Notes as a single user environment while employing multiple applications.

Software Prerequisites

In order to develop and test an LSX with reasonable efficiency using the Toolkit, the following software is required on your workstation:

- One of the following supported development platforms:

  Windows 3.X, Windows 95, Windows NT (on both the Intel and Dec Alpha architectures), OS/2 Warp, HP-UX, Sun Solaris, AIX, Macintosh. At the time of writing this chapter, the HP-UX and Macintosh platforms were not available.

- A standard C++ development environment for the development platform. This includes a C++ compiler, a C++ debugger, the platform’s linker, and a Make utility.

  Note For more detailed information about hardware and software requirements, refer to the *LSX Toolkit Documentation*

- The installed LSX Toolkit.
- The application to be scripted.

Tip When this book was written, version 1.0 of the LSX Toolkit was available, for which it was necessary to disable the compiler feature to treat warnings as errors. Otherwise the compilation process of the LSX samples may abort. You can do so by changing the appropriate compiler macro in the file named LSX_MAKE.{WIN,OS2} in the SRC directory of the distribution.

What the LSX Toolkit Contains

The LSX Toolkit includes:

- C++ base class source code.

  This is intended to be used in your LSX sources.

- Source code for LSX examples.

  There are three working examples, including one that provides a complete LSX template for you to develop an LSX from.

- LotusScript source files.

  These are header files that define the LSX API for the LSX builder.

- Build tools and testing tools.

Included Files

The installed Toolkit has the following directory structure:

```
LSX
 ├── BIN        Utility executables and libraries for all platforms
 ├── INC        Header files for the LotusScript interface API
 ├── LIB        Compiled C++ libraries required for any new LSX
 │              implementation (from sources in "SRC\COMMON")
 ├── SRC
 │    ├──COMMON      C++ source code required for any new
 │    │              LSX implementation
 │    ├──LSXBENTO    Sample: Interactions with Bento Container
 │    │              Manager
 │    ├──LSXTW       Sample: Text window manipulation
 │    │              using the native window system
 │    ├──TEMPLATE    Sample: Template to create new LSXs
 │    └──LODLTEMP    Sample: Automatic creation of
 │                   C++ class method definitions
 ├── TESTS      Test scripts for sample LSXs
 ├── XINC       API header files of the applications wrapped by
 │               the sample LSXs
 └── XLIB       API libraries of the applications wrapped by the
                sample LSXs
```

For the Intel and DEC Alpha platforms, the BIN and LIB subdirectories are further divided into:

- OS2
- W16
- W32
- ALPHA

For UNIX platforms, they are divided into:

- AIX
- HPUX
- SUN
- X86

Each of them contains the executables and libraries for a particular development platform (hidden in the figure). The same applies to the OBJS subdirectories contained in each of the SRC subdirectories. They store the compiled object modules of the sources.

Note If you are developing for a single platform, only one of those platform directories is of interest. On the other hand, if you install the Toolkit on a network file system accessible from multiple development platforms, this directory structure serves as a basis for cross-platform development since you can produce different library formats out of a single source code.

The sources in the SRC\COMMON and SRC\TEMPLATE directories form a C++ class framework in which you can plug the classes intended to be used from within LotusScript.

Utilities for Building and Testing

The C++ build tools comprise makefiles for all supported development platforms as well as front end DO_IT batch-file build utilities.

Certain special LSX build tools are furnished as executable files in the platform-specific subdirectories of \LSX\BIN:

- LSXLODL, a compiler to convert class member declarations in Lotus Object Definition Language to C++ definitions of certain tables that an LSX uses to register with LotusScript.
- LSXTEST, a GUI test frame for writing, running, and debugging LotusScript scripts, including running LSX modules.
- LSXRUN, a command-line test frame for testing LotusScript scripts, including running LSX modules. It does not depend on any graphical user interface.
- LSXREG, for registering your LSX with the platform's class registry.

Considering the Toolkit Design

Apart from the inherent language features of LotusScript, scripts always require an embedding application context like Notes that provides them with "physical" objects to work on. Therefore, the LotusScript instance responsible for compiling and executing scripts contains an open interface to be able to connect to an embedding application. This separation of functionality into embedding and embedded components, and a well-defined interface between them, forms the basis of the LSX integration.

The Extendible LotusScript Architecture

On startup, Notes creates a LotusScript instance for all further script processing.

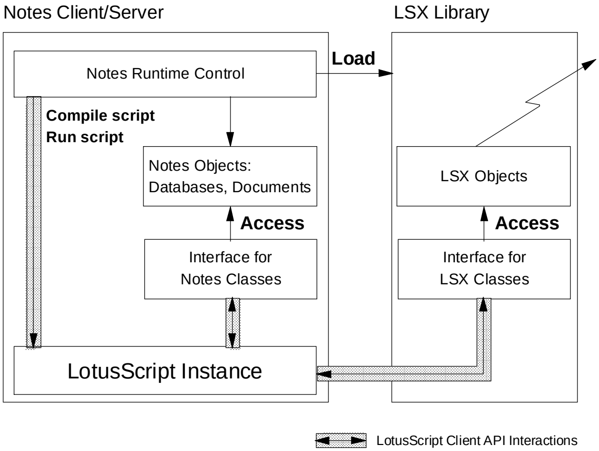
This instance provides certain services via a LotusScript client API which is accessible by an API identifier, referred to as the LotusScript instance handle. Notes gets the handle as a result of the creation.

Notes as the embedding application controls the operation of LotusScript through LotusScript's client API. Notes presents LotusScript source or compiled code to the LotusScript instance via this API, and LotusScript compiles and executes them.

Crucial to the LotusScript architecture is the fact that the LotusScript client API contains services to register new classes. Notes uses these services to register its LotusScript Notes classes.

The implementation of each class is included in Notes' code space: part of the registration function serves to specify the entry points that the LotusScript instance can call to execute the scripted behavior. This means that Notes supplies the LotusScript instance with callback functions that implement class constructors, methods, and property access.

The same method applies to the integration of an LSX module that is compiled and linked as dynamic library. First, Notes loads the library and calls a well-known function entry point in the library with the handle of the LotusScript client API. Now, this LSX function uses that handle to register the LSX classes, including the methods and properties that make up the class definitions. Since the callback functions that implement the registered functions, and the methods of the classes, are also in the LSX library, the LotusScript instance knows how to execute the external implementation.



The following sections describe the interactions between the LSX and the LotusScript instance embedded in Notes.

LSX Integration

After loading the LSX, Notes obtains the address of the LSX message procedure. All further communication between Notes and the LSX happens via this message procedure, defined by a standard set of messages. The two most important messages are INITIALIZE and TERMINATE for LSX enrollment.

Having retrieved the message procedure address in the LSX, the first message Notes sends to the LSX is INITIALIZE, passing the LotusScript instance handle as a parameter.

As mentioned previously, it is possible that an LSX is simultaneously used by more than one Lotus application. For example, if the LSX is a shared library loaded on a multi-user platform like UNIX, its code may be shared between multiple Notes workstations, each of them embedding a LotusScript instance. In that case, the LSX is loaded only once, but it receives multiple INITIALIZE messages indicating the start of a session with a new LotusScript instance. The LSX is responsible to maintain all these sessions to be able to perform a proper cleanup for each of them when a TERMINATE message arrives.

LSX Initialization

In the initialization phase, an LSX must register its classes with the LotusScript instance using the passed handle.

Registering a class means supplying LotusScript with a complete class definition that will enable processing any runtime operations on the class, including creating and destroying class instances (objects). The class-definition information includes the class name, class ID, version number, parent class ID; tables to define the properties, methods, and events of the class; and miscellaneous other information.

Since the implementation of each function or class is in the LSX's code space, LotusScript must call back to the LSX at runtime to create and manipulate instances of that class. So part of registering a class is providing a callback function for the LSX to use at runtime when LotusScript calls back with a request to carry out operations on objects that the running script has specified. For a given class, this function is known as the class control procedure. It must handle the object-manipulation messages sent to it by LotusScript, such as the CREATE message to create an object.

Once the INITIALIZE call returns, the LSX is idle except when it receives a message it must respond to.

Object Creation

When an executing script requests a new instance of an LSX class, the LotusScript instance calls the registered class control procedure for that class to send the CREATE message. After the new object is created, it is added to a particular list containing all objects that were created in the current session.

An object presents itself to the LotusScript instance via an object control interface. LotusScript uses this interface for all further interactions with a new object. It defines a standard set of messages for object method invocation and property access.

Object Deletion

Deletions are handled in a similar manner. The LotusScript instance sends a DELETE message together with an object ID to the appropriate class control procedure which has to delete the object and update the session object list.

Runtime Manipulations on Objects

The object control interface receives messages for method invocation, setting and getting properties, and several other messages. The interface must map the message parameters onto the corresponding LSX class methods and attributes to gain the intended object behavior.

Event Notifications

The LSX class method implementation may raise events to signal special conditions to the executing script. As intended by the LotusScript language, the script can catch them with installed event handlers. Likewise, LSX methods can cause errors to be raised which are then handled in the executing script. The LotusScript client API comprises appropriate functions for that purpose.

Part of the definition for any LSX class that is registered with LotusScript are the events raised (if there are any), and under what conditions they are raised.

LSX Termination

Just before destroying a LotusScript instance in which the LSX is loaded, Notes sends the TERMINATE message to the LSX message procedure. The LSX is responsible for cleaning up any of its objects that belong to that instance.

In order to guarantee that LSX class objects of other LotusScript instances remain valid, only the objects of the current session’s object list may be cleaned up.

Understanding the C++ LSX Class Framework

The LSX Toolkit supplies you with a set of C++ classes and functions that are to be used in an LSX on top of the LotusScript client interface. This code provides higher-level services for the LSX, including class registration utilities, and the infrastructure for handling LotusScript callbacks.

When you develop a new LSX, the Toolkit code forms a framework in the sense that you can reuse its functionality by deriving your LSX classes from Toolkit classes, and by extending the implementation of certain global Toolkit functions. Therefore, you need to know where the supplied classes and functions are located, and how they interact with each other.

Important LSX Source Files

The directory INC contains C header files for the LotusScript client API. In the file INC\LSILSX.H, the C++ struct LSsLsxInstance defines the raw interface through which all communication between the LSX and the LotusScript instance occurs.

The directory SRC\COMMON contains the following files:

- LSXBASE.HPP
- LSXBASE.CPP
- LSXCOMM.HPP
- LSXCOMM.CPP

They have a central role in any LSX built using the LSX Toolkit.

Note You can use the code in this directory without any modification.

The SRC\COMMON\LSXBASE.[CH]PP files constitute an isolation layer within the LSX. LSXBASE.HPP defines the LSXBase base class, an abstract C++ class that every class in your LSX should inherit from. LSXBASE.CPP implements the base class. This LSXBase class serves mainly as an interface class; it comprises the object control interface (by which LotusScript accesses the LSX class objects), and provides your classes with an easy callback mechanism. Moreover, it performs some of the object protocol messages automatically, and it contains a linked list implementation for maintaining a hierarchy of LSX objects.

The files SRC\COMMON\LSXCOMM.[CH]PP constitute most of the interface between the LSX and Notes and its embedded LotusScript instance. They provide essential services for the LSX, such as an implementation for the LSX message procedure, a generalized class control procedure, and registration utility functions.

The directory SRC\TEMPLATE contains the following files:

- LSXSESS.HPP
- LSXSESS.CPP
- LSXSESS.TAB
- OTHER.HPP
- OTHER.CPP
- OTHER.TAB

They form an actual application, and are intended to be used as a template for your LSX development.

Note You will need to modify this code for your LSX.

The files SRC\TEMPLATE\LSXSESS.[CH]PP define and implement the class LSXSession. As described previously, it is possible that an LSX is connected to multiple LotusScript instances at a time. For each connection, a single LSXSession object maintains information about the objects created in it, to ensure a proper session cleanup. Moreover, the file LSXSESS.CPP contains the (LSX specific) class registration code.

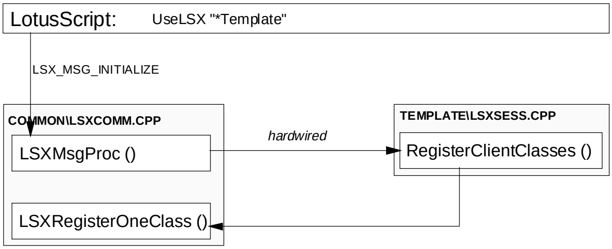
The files SRC\TEMPLATE\OTHER.[CH]PP define and implement a sample LSX class OTHER derived from LSXBASE.

The files LSXSESS.TAB and OTHER.TAB define static tables containing the required type information for the LSX class registration. They are automatically generated with the build tool LSXLODL.

Flow of Control Within the Framework

Now, let's consider how the Toolkit framework implements the interactions with the LotusScript instance. The main LSX tasks such as initialization, object creation, and object manipulation, are described from an implementation point of view. You will find the LSX specific code sections in the SRC\TEMPLATE files that you have to modify for your LSX.

After being loaded by Notes, an LSX registers its LSX message procedure LSXMsgProc located in SRC\COMMON\LSXCOMM.CPP. Then, the LotusScript instance calls that function with an INITIALIZE message parameter:

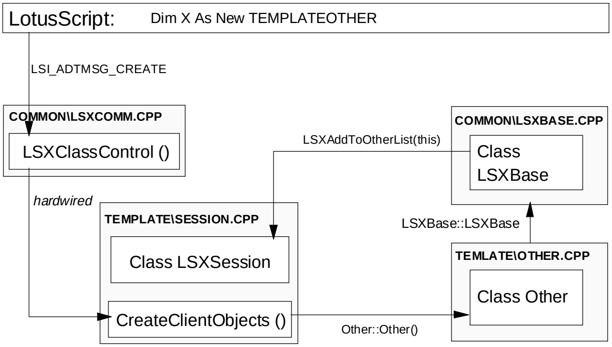


In order to register all LSX classes, the message procedure calls the function RegisterClientClasses. This LSX specific function knows the LSX classes, namely LSXSession and Other, retrieves the type information from the static tables in the .TAB files, and registers each of them with a separate call to the function LSXRegisterOneClass. Eventually, this utility function uses the LotusScript client API to perform the registration.

Note To register your LSX classes, you have to modify the function RegisterClientClasses.

Part of a class registration is to provide a class control procedure which the LotusScript instance uses to execute class operations. The Toolkit includes a generic function LSXClassControl that can be used for all LSX classes. The function LSXRegisterOneClass registers it as the related callback function.

This callback function is used when LotusScript encounters a New statement for an LSX class in an executing script.



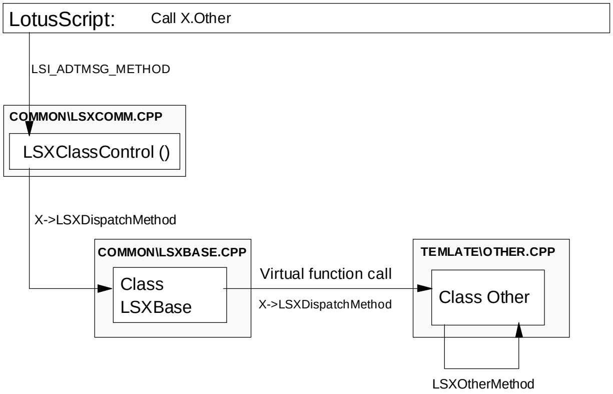
The LotusScript instance calls LSXClassControl with the message parameter LSI_ADTMSG_CREATE and the ID of the LSX class. Because class object creation requires LSX specific knowledge, it simply passes the call to the function CreateClientObjects.

Now, this function decides on the given class ID what kind of object is to be created, and calls the C++ new operator for this LSX class. As usual with C++, the constructor of that class first calls the constructor of the base class (which is always LSXBase). The base class constructor now registers itself with the session object in that it is created. Furthermore, it saves the given LotusScript handle as an object attribute so that the LSX class object can use the LotusScript API later on. Eventually, the body of the LSX class constructor can implement application specific object initialization as needed.

When the new LSX class object is created, the function CreateClientObjects returns a handle to the object control interface to the calling class control procedure.

The LotusScript instance now uses this handle to manipulate the object. In fact, the class LSXBase comprises the object control interface, because it is derived from it. Again, LotusScript accesses this interface by a callback function, and the Toolkit design strategy is to use the same control procedure for objects as for classes. This means that the LotusScript instance finally calls the function LSXClassControl to perform object manipulations, passing the message together with a class ID, an object handle, and a set of parameters.

Let us now consider an example: a method invocation on a given object.



Receiving an LSI_ADTMSG_METHOD message, the LSXClassControl function first converts the given object handle into an LSXBase object. Then, it calls the method LSXDispatchMethod on that object which is defined as pure virtual function in LSXBase. Therefore, it actually calls the function defined in the derived class Other. Now, this function determines which method invocation is requested (by looking at the method ID), calls it, and passes the return value back to the caller.

As you can see, method invocation on objects is a very straightforward implementation. It takes advantage of the class LSXBase which is designed as an interface class. Other runtime manipulations occur in the same manner. For example, a script statement to access an Other object property is sent as a LSI_ADTMSG_PROP_GET message to the function LSXClassControl which calls the method LSXGetProp on the LSXBase object. Again, the function is declared as virtual, and the object is actually of class Other, so that the function LSXGetProp in the Other class is called. Finally, this function is provided with the property ID, and can take the appropriate action.

Other interactions between the LotusScript instance and the LSX, such as object deletion and LSX termination, are almost done automatically. The implementation of the LSXSession class ensures a proper cleanup per session, and calls the destructor of any other LSX class objects as needed.

LSX Design Decisions

In the following, find some general design decisions to be considered for all LSX implementations.

LSX Class Design

You need to consider what kinds of data structures to use to represent the object attributes in your class model.

Besides scalar types such as INTEGER, LONG, SINGLE, DOUBLE, CURRENCY, STRING, or VARIANT, LotusScript supports arrays and lists. Beyond it, you can use any of the classes you define in the LSX. All of these data structures are available for declaring and using as data members of your LSX classes. The same holds for the parameters and return value types of the class methods.

Note This implies that you cannot directly interface to the Notes product classes such as NotesDocument. You have to break them down to the types LotusScript supports.

A further way to structure an LSX class is to define and register it as a collection class. A collection class is a container of items that can be accessed directly via indexing or via the LotusScript ForAll language iteration construct. The allowable language constructs are to access the values, the properties, and the methods of an individual item, or of every item in the collection.

Object Control Interface

LotusScript follows the conventions of a COM (Common Object Model) interface in accessing client objects. It is the object control interface, a C++ structure named ILsiADTControl.

Either the ILsiADTControl structure may be contained as a member in the definition of each class, or the base class LSXBase may inherit from it. The usual design strategy, and the default in the Toolkit examples, is inheritance.

In future releases, it is planned to include an OLE adapter in the Toolkit enabling you to expose objects in any of your LSX classes to OLE automation. This technique will require the ILsiADTControl to be inherited. For now we recommend using the default, letting the LSXBase class inherit from ILsiADTControl.

In the Toolkit, the high-level flag variable EMBED_ADT governs the choice. It is referenced when building object files for the example LSXs, using the makefiles which are included. By default, the EMBED_ADT flag is undefined, so that the example LSXs compile with ILsiADTControl inherited.

Character Sets

Another design decision is what character set to use to represent LSX-maintained strings that must be passed to the LotusScript instance.

The LotusScript internal representation is UNICODE. However, an LSX or an embedding application can specify any of four string communication representations to LotusScript:

- The platform-native character set (currently ANSI)
- UNICODE
- LMBCS
- ASCII

This means that a string will be presented to LotusScript in that representation. LotusScript is responsible for converting the string to UNICODE as needed for its own purposes.

Caution LotusScript will translate string message parameters passed to the LSX into the representation specified during the class registration. The current implementation of the utility function LSXRegisterOneClass specifies the platform-native character set which is sufficient for many but not all applications.

An LSX cannot specify that every string passed between it and LotusScript uses one of the four representations. Rather, the LSX has to specify which representation to use for each string individually.

Portability Issues

You need to decide early on whether your LSX is to be written for one platform or several. Single-platform design allows you to write C++ source code to take advantage of specific compiler features and system services. However, the resulting source code may not be portable.

In the LSX Toolkit, the provided C++ framework code is platform-independent concerning compiler features and system services. A platform-specific header file is selected and included in those files. The selection criterion differentiates the platforms 16-bit Windows, 32-bit Windows, OS/2, UNIX, and Macintosh.

The LotusScript instance offers you standard systems services: memory management, file management, national language string support, interprocess communication, dynamic library system, and others. Your LSX implementation should access these system services only through the provided LotusScript interface.

Caution The Toolkit overrides the default C++ new operators to use LotusScript's memory management services.

Graphical User Interface

As a separately loaded library, you can develop the LSX to present its own user interface. However, any such interface that you may choose to implement is independent of Notes, and you cannot build interactions between them. In particular, an LSX running on a server cannot invoke it.

Globally Unique IDs for LSX Classes

The client object interface is standardized as an OLE2/COM-style interface. This ensures that client objects are accessed consistently across LotusScript applications.

So, for each class in your LSX, you have to assign a globally unique ID (GUID) to identify your class with LotusScript. LotusScript will not allow an LSX to register a class that has the same GUID as an already-registered class.

A GUID is a 16-byte globally unique identifier. In Windows, a GUID is the same as a Windows GUID used for OLE objects. Some compilers on Windows platforms include a tool to create GUIDs. For detailed information, refer to the *LSX Toolkit Documentation*.

Creating an LSX

This section describes how to create a new LSX.

Currently, setting up the environment for a new LSX is a task where you have to create several files for the new LSX classes. Beyond that, you have to make some changes in the files that comprise the Toolkit, namely LSXSESSION.HPP and LSXSESSION.CPP.

Note This means the first thing you want to do before starting to implement an LSX, is to copy the sources from SRC\TEMPLATE to a new source directory SRC\NEWLSX for your LSX. Then, change the SUBSYS entry in the MAKEFILE.MAK and the Library entry in TEMPL_[NW].DEF to your LSX name. On Windows, the files LINKRESP.W32 (for 32-bit) and OBJSRESP.* (for 16-bit) must contain the appropriate directory.

Caution Also, be sure to change every occurrence of “Template” to the name of the LSX. It is very important that propercase be maintained where applicable. For example: “Template” to “Newlsxname” and “TEMPLATE” to “Newlsxname.”

Follow these steps to set up a new LSX class:

1. For each new LSX class, define the properties, methods, and events it shall have. These definitions are contained in several tables (C++ structure arrays) and constants, stored in a .TAB file. To define them, you can either directly start with a copy of the OTHER.TAB file and modify it, or you first define them in the Lotus Object Definition Language, and use LSXLODL to compile them into those tables. Then, create an .HPP file for the class definition, and use the file OTHER.HPP as an example of how to structure it. The same applies to the .CPP file you create for your class. At least, your class must contain a constructor, a dispatch method, and two more methods to get and set properties.
2. Next, modify the files for the LSXSession class (in the new directory). Apart from some changes to constants and tables for GUIDs and names, you must add another list to maintain all objects of the new class as well as methods to add and delete items in it. Furthermore, update the registration function to register the new class, and extend the constructor and destructor to set up the new list member properly.
3. Then, update the file TEXTSTR.HPP to assign IDs to the methods, properties, and events. In the file GUIDFILE.HPP, define a GUID for your class. You can always use the existing code as a guideline. It contains comments that suggest where to put the new code.
4. Finally, make the appropriate changes to the makefile in order to compile your LSX. If you are developing on a Windows 3.X platform, you also have to add the path of the new LSX object to be created.

Tip If you don't want to implement all the functions at once, you can still build the LSX and use it in your scripts. For all the functions not implemented yet, simply return the value LSI_RTE_SubOrFunctionNotDefined from within the class method LSXDispatchMethod so that LotusScript will notice it.

An Example

Here is an example of how to add a method NewMethod to the class Other which has no return value and a single INTEGER argument. Start by setting up a new LSX source directory as described, and do the following:

1. Changes in OTHER.TAB:

   Extend the method ID table:

   ```
   static LSUINT other_methodnameids[N_OTHER_METHODS] =
   {
      CMYLSX_OTHERMETH_NEW,
      CMYLSX_OTHERMETH_CLOSE,
      CMYLSX_OTHERMETH_OTHER,
      CMYLSX_OTHERMETH_PASSOBJ,
      CMYLSX_OTHERMETH_NEWMETHOD, // the new entry!
   };
   ```

Create the list of arguments that this method will be receiving:

```
static LSDATATYPE
NewMethodArgs[N_NEWMETHOD_METHOD_ARGS+1] =
{
   LSX_BYREF_VOID,   // return type
   LSX_THIS_PTR,     // ptr to created instance
   LSX_BYVAL_SHORT   // the INTEGER argument
};
```

Create a method description used at registration time:

```
static LSADTMETHODDESCR
other_gmethods[N_OTHER_METHODS] =
{
   //... all previous entries
   { (LSPLTSTR)LSNULL, CTEMPLATE_OTHERMETH_NEWMETHOD,
     NewMethodArgs,
     (LSREGNAME*)LSNULL,N_NEWMETHOD_METHOD_ARGS,
     LSX_REGULAR_PROC,LSI_NO_HELPID, 0 },
};
```

2. Changes in OTHER.HPP:

   Change the number of methods:

```
#define N_OTHER_METHODS          5
```

   Add the definition for the number of method arguments:

```
#define N_NEWMETHOD_METHOD_ARGS  2  // 1 + this
```

   Add the method to the class declaration:

```
class Other : public LSXBase
{
   public:
      //...
      void NewMethod (PLSADTMSGMETHOD args);
      //...
};
```

3. Changes in OTHER.CPP:

   Extend the dispatcher method:

```
LSSTATUS Other:: LSXDispatchMethod (PLSADTMSGMETHOD args)
{
   //...
   switch (args->idMeth)
   {
      //...
      case CTEMPLATE_OTHERMETH_NEWMETHOD :
         this->NewMethod (args);
         break;
      default:
      //...
}
```

Define the new method: (it will be more useful later in this chapter)

```
void Other:: NewMethod (PLSADTMSGMETHOD args)
{}
```

4. Changes to LSXSESS.CPP:

Update the LSX name table:

```
static TEXTTABLE gTemplateNames[] =
{  //...
   {CTEMPLATE_OTHERMETH_PASSOBJ,   "PassObject"},
   {CTEMPLATE_OTHERMETH_NEWMETHOD, "NewMethod"},
   //...
```

Note The table entries occur in the following order: first class names; then the properties, methods, and events for the first class; then for the second; and so on.

5. Changes to TEXTSTR.HPP:

Define the ID for the new method:

```
//...
#define CTEMPLATE_OTHERMETH_OTHER    (LSXBASE_NAMES+482)
#define CTEMPLATE_OTHERMETH_PASSOBJ  (LSXBASE_NAMES+483)
#define CTEMPLATE_OTHERMETH_NEWMETHOD(LSXBASE_NAMES+484)
//...
```

Assuming that you have set up all the files for a new LSX, you may want to add the specific application logic. The next sections explain how to use LotusScript data types, variables and method arguments of these types, and the system services offered by the LotusScript instance.

Using LSX Data Types

The LSX interface headers define macros for common C++ data types to compensate platform differences. You are encouraged to use them instead of the intrinsic C++ language types, because they are widely used throughout the Toolkit source code, and you should be familiar with them.

Data Type Macro	*Meaning*
LSVALUE	Any of the following data type macros but the unsigned versions; actually a union of all of them plus a member Type that tells you the type of the value. (See the section on Data Type Descriptions below)
LSSBYTE	Same as signed char
LSUBYTE	Same as unsigned char
LSSSHORT	Same as INTEGER in LotusScript and signed short int in C++

continued

Data Type Macro	*Meaning*
LSUSHORT	Same as unsigned short int
LSSINT	Same as signed int
LSUINT	Same as unsigned int
LSSLONG	Same as LONG in LotusScript and signed long in C++
LSULONG	Same as unsigned long
LSFLOAT4	Same as float
LSFLOAT8	Same as double
LSBOOL	Allowed values: LSTRUE, LSFALSE
LSPVOID	Pointer to void (null pointer is LSNULL)
LSPBYTE	Pointer to (unsigned) character arry
LSPTR (x)	Pointer to x
LSSTATUS	Same as LSSSHORT; used for return status values

Some more type macros are defined for string data types:

Data Type Macro	*Meaning*
LSCHAR	Same as char
LSPLTCHAR	Same as LSCHAR
LSCLICHAR	Same as LSCHAR
LSUNICHAR	Same as LSUSHORT
LSSTRING	Unicode string; pointer to LSUNICHAR
LSCLISTR	ANSI strng; pointer to LSCLICHAR
LSPLTSTR	Platform-native string; pointer to LSPLTCHAR
LSUNISTR	Unicode string; pointer to LSUNICHAR

Finally, the remaining data structures defined in LotusScript are represented by the following C++ type definitions:

C++ Data Type Definition	***Meaning***
LSsDate	Variant of data type 7 (as returned by the LotusScript function "Date"); same as LSFLOAT8: For an explanation, refer to the *LotusScript Language Reference Manual*

struct LSsCurrency {
  unsigned long   Lo;
  long   Hi;
}
	Currency data type in LotusScript

PLSVALUE	A variant; same as LSPTR (LSVALUE)

Using Data Type Descriptions

As you may have already noticed, the argument passing to LSX class methods is different from the way C++ passes arguments. Instead, an LSX class method receives a packed array structure as a single argument containing the actual parameters of the method. Each of the array members stands for one parameter, and includes information about the data type and the value.

So, the data type information itself is coded using certain symbolic integer constants. The basic ones are as follows:

LotusScript Data Type Code	***Related Data Type***
LSVT_EMPTY	EMPTY value for variants
LSVT_NULL	NULL value for variants
LSVT_SHORT	LSSSHORT (Integer data type in LotusScript)
LSVT_LONG	LSSLONG (Long data type in LotusScript)
LSVT_SINGLE	LSFLOAT4
LSVT_DOUBLE	LSFLOAT8
LSVT_CURRENCY	LSsCurrency
LSVT_DATE	LSsDate

continued

LotusScript Data Type Code	*Related Data Type*
LSVT_STRING	Depends on the specified character translation! Either a LSSTRING for UNICODE, or LSPBYTE for LMBCS, or LSPLTSTR for ANSI
LSVT_BOOLEAN	boolean
LSVT_VARIANT	PLSVALUE
LSVT_UNISTR	LSsValueUniStr; same as LSPTR (LSUSHORT)

Furthermore, there are constants for list and array type descriptions. Refer to the file INC\LSIVAL.HPP and the *LSX Toolkit Documentation* for details.

Accessing LSX Class Method Arguments

LSX class method arguments are packed in a single array which is passed to your method implementation. So, any of your methods will look like this:

```
void NewClass:: NewMethod(PLSADTMSGMETHOD args)
```

The type PLSADTMSGMETHOD, defined in INC\LSILSX.HPP, is a pointer to the following structure:

```
struct LSFAR LSADTMSGMETHOD
{
   PLSVALUE        pArg;      // Array of Arguments.
   LSUSHORT                     nArg;      // Number of
Arguments.
   LSADTMETHODID   idMeth;    // Method ID.
   LSADTCLASSID    idClass;   // class ID for method.
};
```

```
typedef LSPTR (LSADTMSGMETHOD)    PLSADTMSGMETHOD;
```

The idClass and idMeth members inform you about the class ID and the method ID, respectively. The nArg member tells you the size of the pArg which is actually an array. So, pArg has members from index 0 to nArg - 1. The size depends on whether the method returns a value or not. If it does not, nArg equals the N_**NEWMETHOD_NEWCLASS**_ARGS -1, where N_**NEWMETHOD_NEWCLASS**_ARGS +1 is the number of method arguments you specified in the **NewClass**.TAB file. If the method does return a value, nArg is equal to this constant.

So, basically you access the message parameters by accessing the pArg member. It is an array whose members can be of any type, as defined by the type LSVALUE. But LSVALUE provides you with information about the type of value, and therefore you should always access a method parameter in the following way:

```
// this method doesn't return a value, so it's first
// parameter is at index 0
void NewClass:: NewMethod(PLSADTMSGMETHOD args)
{
   PLSVALUE                pVal = LSNULL;
   // check that the number of passed parameters equals
   // the expected number (means: the declared number of
   // arguments). The constant N_NEWMETHOD_NEWCLASS_ARGS
   // should be defined in NewClass.HPP
   assert (args->nArg == N_NEWMETHOD_NEWCLASS_ARGS - 1);
   //...

   // access the n-th parameter which should be of type
   // INTEGER, for example.
   pVal = &args->pArg[n - 1];
   assert(pVal->Type == LSVT_SHORT);

   // now, it is safe to access the value of the parameter
   LSSHORT nthParm = pVal->vShort;
}
```

Note To apply this method to a class constructor, simply consider it as a method with no return value. So, the first argument starts at index 0.

In this example, the method parameter is accessed by the expression pVal->vShort. In general, to access any of the types whose descriptions are listed in the previous table, the following definition of the data type LSVALUE applies:

```
struct LSsValue
{
   union
   {
      LSSSHORT      vShort;        // LSVT_SHORT
      LSSLONG       vLong;         // LSVT_LONG
      LSFLOAT4      vSingle;       // LSVT_SINGLE
      LSFLOAT8      vDouble;       // LSVT_DOUBLE
      //...
      LSsCurrency   vCurrency;     // LSVT_CURRENCY
      LSsDate       vDate;         // LSVT_DATE
      LSSTRING      vString;       // LSVT_STRING
      //...
      LSsValueBool  vBool;         // LSVT_BOOLEAN
      PLSVALUE      vVar;          // LSVT_VARIANT
      //...
      LSsValueUniStr    vUniStr;     // LSVT_UNISTR
      // —- Convenience Values for callbacks
      //     when translation is specified
      LSPBYTE       vLmbcs;         // LSVT_STRING
                                    // (translated)
      LSPLTSTR      vChars;         // LSVT_STRING
                                    // (translated)
      LSPBYTE       vBytes;         // LSVT_STRING
                                    // (translated)

      //...
   }; // end of union
   LSVALTYPE   Type;   // Value Type
   //...
};
```

Furthermore, there are members for the types such as lists and arrays. Refer to the file INC\LSIVAL.HPP and the *LSX Toolkit Documentation* for details.

LSX Error Values

The data type LSSTATUS is frequently used as a return type, to either return LSX_OK, or any of the error constants defined in the file INC\LSIERR.HPP.

Accessing LSX Class Property Arguments

For each class with properties exported to LotusScript, you will set up two class methods **NewClass**::LSXGetProp and **NewClass**::LSXSetProp. For the latter, the question arises how to access the new value that the property should get.

The declaration is as follows:

```
LSSTATUS NewClass:: LSXSetProp(PLSADTINSTDESC pInstDesc,
                               PLSADTMSGPROP  param)
```

The first argument, pInstDesc, is a pointer to a structure describing the called object in terms of object control interface, related class ID, and current LotusScript instance.

The second argument is more important since it names the property to be changed, and the new value. It points to the following structure:

```
struct LSADTMSGPROP
{
   PLSVALUE         valProp;   // Property Value.
   LSADTPROPID      idProp;    // Property Id.
   LSADTCLASSID     idClass;   // Class ID for this property.
};
```

The member idProp stores the ID of the property to be changed, as you registered it in the .TAB file. The valProp member contains the new value of the property. You use this structure as follows:

```
LSSTATUS NewClass:: LSXSetProp(PLSADTINSTDESC pInstDesc,
                               PLSADTMSGPROP param)
{
   LSSTATUS               stat = LSX_OK;
   PLSVALUE   pVal = param->valProp;
   LSSHORT    len;

   switch (param->idProp)
   {
      case CTEMPLATE_NEWCLASSPROP_PROPERTY:
         // access property (assuming it's a
         // LotusScript integer)
        len = pVal->vShort;
        //...
     default:
        assert (LSFALSE);
     }
   return stat;
}
```

Using LotusScript System Services

The LotusScript client API offers you several system services. You are encouraged to use them rather than directly accessing the operating system API. This helps you to write LSXs which can be ported to other platforms more easily.

The following paragraphs give you an overview of some of the services. For a detailed description, refer to the files contained in the directory INC\SYS. For example, the file management service is declared in the file LSSFMGR.HPP; the actual service is enclosed in SERVICE_DECL_BEGIN(FILEMGR) and SERVICE_DECL_END. It consists of a set of functions you may call in any of your class methods.

For any but the memory management system service, you have to prepare your class to use the service.

Preparational Steps

1. First, extend your LSX class slightly by adding a new private member:

   ```
   class NewClass : public LSXBase
   {
         //...
      private:
         //...
         PLSSFILEMGR   pFM;  // system services file manager
         //...
   };
   ```

   The member type is taken from the file INC\SYS\LSSFMGR.HPP.

2. Then, extend the class constructor to initialize it:

```
NewClass:: NewClass (LSPTR(LSXSession) s,
                     PLSADTMSGCREATE args)
   : //... initialization of base and class members
{
   LShINSTANCE   hLSInstance; //  LS instance handle for
                               //  this class object

   // get the instance
   hLSInstance=s->LSXGetInstance();

   // get the file manager service handle
   this->pFM = hLSInstance->Services->pFMGR;
   //...
}
```

You will find a complete list of available system service handles in the file INC\LSSRVMGR.HPP. They are defined as members of the structure LSSsAnchor.

3. Now, your class is ready to access the service in its methods:

```
void NewClass:: NewMethod (PLSADTMSGMETHOD args)
{
   assert (this->pFM);   // check that the file service
                         // is available

   //...
   this->pFM->"any FM service function"
}
```

Memory Management Service

The API provides function calls for allocating and releasing heap memory. When the LSX uses this service, the LotusScript instance gains complete control over the dynamically allocated memory.

In order to make use of C++'s ability to redefine the new operator, the file SRC\LSXCOMM.CPP defines a special version of operator new, operator new [], and a corresponding delete operator that perform calls to the API's memory service functions. This means that you don't have to know the API memory functions; you will use them implicitly by these C++ operators.

In contrast to the ordinary new operator, the versions defined for LSXs get a so-called placement argument, a handle to the LotusScript client API. The following code fragment shows you how to use them:

```
void NewClass:: NewMethod(PLSADTMSGMETHOD args)
{
    LSPTR (LSSSHORT) *aNewInt;
    LSPLTSTR  *aNewString

    // create a new integer (always pass this->LsiInst)
    aNewInt = new (this->LsiInst) LSSSHORT;
    // create a new string of length 42 (incl. trailing 0)
    aNewString = new (this->LsiInst) LSPLTCHAR [42];
    //...
    // do something
    //...
    delete aNewInt;
    delete aNewString;
}
```

File Management Service

The LotusScript client API offers you a rich set of file management functions:

- Functions for file access: Open, PathOpen, Close
- Functions to work on file contents: Read, Write, Seek
- Functions for file attributes: GetAttr, SetAttr, DateTime, FileSize
- Directory functions: ChDir, CurDir, MkDir, RmDir, DirFirst, DirNext
- and more...

The following code gives you an example how to use these service API functions:

```
void NewClass:: NewMethod(PLSADTMSGMETHOD args)
{
   // Purpose: Open a file, append a text string, and close
it

   // set the open flags: lock and access type
   LSUSHORT LSOpenMode = (LASFM_SHARE_EXCLUSIVE |
                          LASFM_ACCESS_WRITE);
   lfile hFile; // a file handle

   // try to open the file; it fails when it doesn't exist
   hFile = pFM->Open ("\\lsx\\newmthd.txt", LSOpenMode);

   if (hFile < 0)  // it has to be created
      hFile = pFM->Open ("\\lsx\\newmthd.txt",
                         LSOpenMode | LASFM_ACCESS_CREATE);

   // seek to the end of the file
   if (pFM->Seek (hFile, 0, LASFM_SEEK_EOF) == LASFM_ERROR)
   {
      // perform error handling!
   }

   LSUSHORT charsWritten;

   // now write the text string...
   charsWritten = pFM->Write (hFile,
                              "A text string", 13);
   // ...and append a newline
   charsWritten += pFM->Write (hFile,
                              LASFM_EOL, LASFM_EOL_LEN);
   // check that all characters are written
   if (charsWritten != 13 + LASFM_EOL_LEN)
   {
      // perform error handling!
   }

   // close the file
   pFM->Close (hFile);
}
```

Interprocess Services

This service is defined in the file INC\LSSIPC.HPP. It offers some functions you may already be familiar with from script programming in Lotus Notes.

- SendKeys
- SendKeysCancel
- Shell
- AppActivate

Platform Services

The platform service definition gives you access to some system values and functions, for example:

- GetDate/SetDate for handling the system date
- GetTime/SetTime for handling the system time
- Environment to retrieve system environment variables
- MsgBox to display texts in a message box

The following example demonstrates the usage of the MsgBox function. Of course, the class must be prepared to access the platform service anchor:

```
void NewClass:: NewMethod(PLSADTMSGMETHOD args)
{
   // Purpose: Display a text in a message box.

   // The box shall consist of an information icon
   // and Yes and No buttons
   LSUSHORT button;

   button = pPLAT->MsgBox ("Do you want to see more?",
                           4 + 64,  // same as MessageBox !
                           "Please decide");
   switch (button)
   {
   case 6: // YES
      //...
   case 7: // NO    (same return codes as for MessageBox)
      //...
   }
}
```

Testing an LSX

During development, to test your new LSX you can either write event scripts in a Notes database that use the classes, or you can use a test tool shipped with the LSX Toolkit.

Because LSX testing isn't concerned with any of the Notes functionality, it is much more convenient to use the Toolkit test tools.

The LSXTEST Tool

In general, LSXTEST presents an integrated development environment to write, compile, execute, and debug LotusScript programs.

In particular, it helps you to write and debug LSX test scripts that contain a USELSX statement to load the LSX.

For example, LSXTEST allows you to:

- Open, edit, and save LotusScript .LSS files.
- Compile scripts and save the compiled LotusScript modules.
- Load compiled modules.
- Set breakpoints to interrupt script execution.
- View the values of variables and the stack frame for the current breakpoint during execution.

Many of the features are also available as command line options passed to LSXTEST. Hereby, you can automate most of the test steps.

Let’s Look at an Example

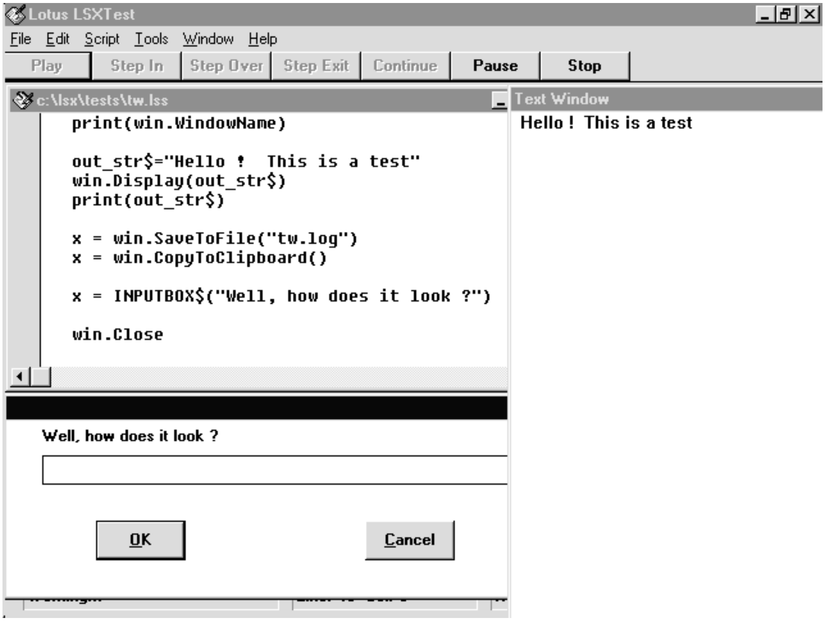
The Toolkit contains test scripts for all sample LSXs. To load and run one of them in LSXTEST:

1. Start an LSX development session with a command prompt window.
2. Start LSXTEST. Actually, the command name is different for each platform. For example, in OS/2, it’s LSXTESTO.
3. Choose File - Open. The file dialog box is displayed.
4. Select one of the test scripts in the directory TESTS, for example TW.LSS. The script is displayed in a new window.

   Note This sequence of steps is also accomplished by invoking LSXTEST with the script filename.
5. Click the Play button to run the script. It uses the Textwindow sample LSX to display a new window.

   Note Remember to first compile the LSX in the directory SRC\LSXTW.

The following figure shows the result:



```
print(win.WindowName)

out_str$="Hello !  This is a test"
win.Display(out_str$)
print(out_str$)

x = win.SaveToFile("tw.log")
x = win.CopyToClipboard()

x = INPUTBOX$("Well, how does it look ?")

win.Close
```

Hello ! This is a test

Well, how does it look ?

The LSXRUN Tool

LSXRUN provides a minimal runtime environment for testing scripts. It is invoked by a command line, and does not require or depend on any graphic user interface.

LSXRUN runs LotusScript source files, and outputs a report of its activities to the screen and an optional log file. For further details, refer to the *Toolkit Documentation*.

Deploying an LSX

The LSX Runtime Environment

The runtime environment of an LSX consists of the following:

- Lotus Notes Release 4. Actually, it can be any Lotus product that supports the LotusScript interpreter (Release 3.0 or higher).
- The LSX itself.
- The scripted application: the application for which the LSX provides an object model.

You must be aware of the location where the LSX will eventually execute. If you define an agent that runs on a server, and it uses the LSX, the location of the LSX is the Notes server. There are other types of agents, for example “Manually from Actions Menu,” that will run on the Notes Workstation. If these agents use an LSX, its location is the Workstation.

So, the LSX and the scripted application must be available at the location where they are used.

LSX Installation

To distribute your LSX, you may want to write an installation program that copies the LSX (and probably the scripted application) to the desired directory, and performs some necessary initialization tasks such as setting up environment variables.

The common method to write an installation program is a batch command file. As your LSX is intended to run on multiple platforms (ideally on all Notes platforms), the installation program should consider the operating system used for installation. This allows the installation to behave differently for different platforms. Look at the Toolkit installation program for examples of how to differentiate between operating systems.

LSX Registration

The registration of the LSX classes influences the method used for accessing them from within LotusScript. Either the script loads it by passing a complete path to the UseLSX statement, or it just references a name in the LSX class registry.

If you choose the first option, your installation program must copy the LSX library to the location referenced by the scripts. In fact, it is very difficult to find pathnames for libraries that consider multiple operating systems, the different directory structures and drive names, and filename restrictions.

Therefore, the class registry is the recommended place to store a complete path of the LSX library, together with a symbolic name, the key, that you then use in the UseLSX statements in your scripts. The key uniquely identifies your LSX on the system. Using this option, LotusScript will look up the path in the LSX class registry.

Be aware that you cannot redistribute Lotus' registration program LSXREG to call it from the installation program. Instead, you have to write your own registration procedure for your classes.

On Windows 3.X, the registry is stored in a LotusScript Extension section in the .INI file. On Windows 95 and Windows NT, the information is stored in the folder HKEY_LOCAL_MACHINE\SOFTWARE\Lotus\Components\LotusScriptExtensions\2.0. On all other platforms, the file NOTES.INI is used as class registry.

Chapter 8
Using Agents

About Agents

Agents allow you to automate many tasks within Notes. They operate in the background to perform routine tasks automatically for the user, for example filing documents, sending mail, looking for particular topics, archiving older documents, or perform more powerful functions, such as manipulating field values and bringing data in from other applications.

Agents can either be private agents created by the user and used only by the user, or shared agents created by a designer and used by anybody who has access to the application or database. Both private and shared agents are design elements that are stored with the database for which they are created. They can be run manually by the user, automatically when certain events occur such as mail arriving, or scheduled to run at certain intervals. They can contain Notes simple actions, @Function formulas, or a LotusScript program.

Access Control

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

In order to create a personal agent, you must have this option selected.

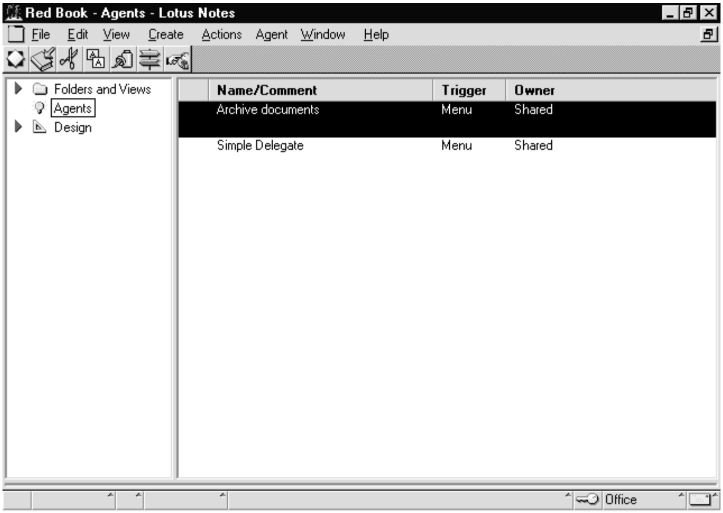
In order to create a shared agent, you require at least designer authority.

Note A Notes administrator can use the Agent Manager Restrictions section of a server document to prevent people from running personal agents on a server; people denied this server access cannot create personal agents on the server, regardless of the ACL setting.

Creating an Agent

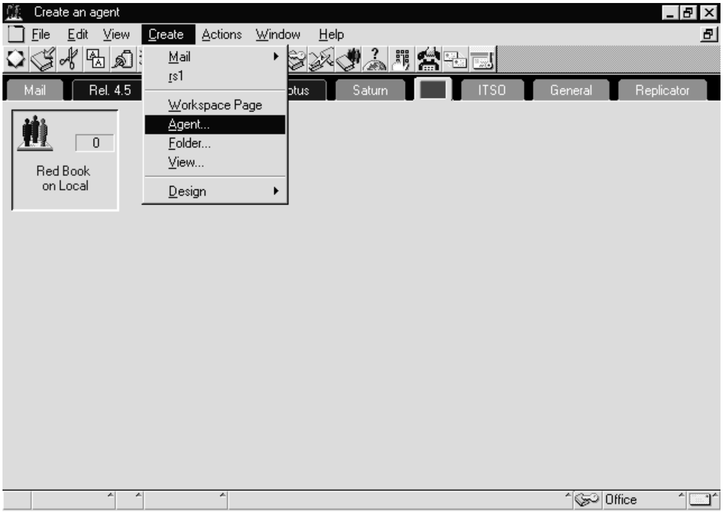
There are several ways of creating an agent.

1. To view existing agents, open the database.
2. Click on Agents within the navigation pane.
3. Double-click on one of the agents displayed on the right-hand side to open it.


Red Book - Agents - Lotus Notes
File Edit View Create Actions Agent Window Help
Folders and Views
Agents
Design
Office

Name/Comment	Trigger	Owner
Archive documents	Menu	Shared
Simple Delegate	Menu	Shared

4. Or, click on the Database icon.
5. Choose Create - Agent...

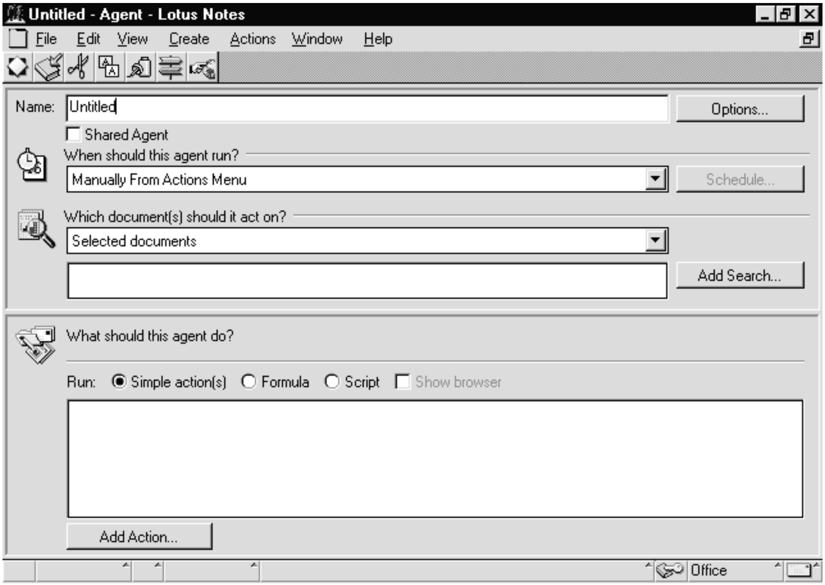


Or, copy and paste from another database. To do this, open the agents pane in the database you want to copy from.

6. Select the agent you want to copy, and copy it to the clipboard.
7. Go to the agents pane in the database you want to copy to.
8. Paste the agent in from the clipboard.
9. Double-click on the agent to work with it.

Setting Up the Agent

Whichever method you use, the following Agent Builder window is displayed:



Name: Untitled

□ Shared Agent

When should this agent run?

Manually From Actions Menu

Which document(s) should it act on?

Selected documents

What should this agent do?

Run: ● Simple action(s) ○ Formula ○ Script □ Show browser

Naming the Agent

The first thing to do is to give the agent a name. A descriptive name is especially important for an agent that you are designing for users to select from the Action menu. Also try to keep the first character unique. This is because, as with forms and views, Notes will use the first unique character as an accelerator key under OS/2, Windows, and UNIX (Macintosh does not use accelerator keys).

Also, click Shared Agent if you want this agent to be used by other users.

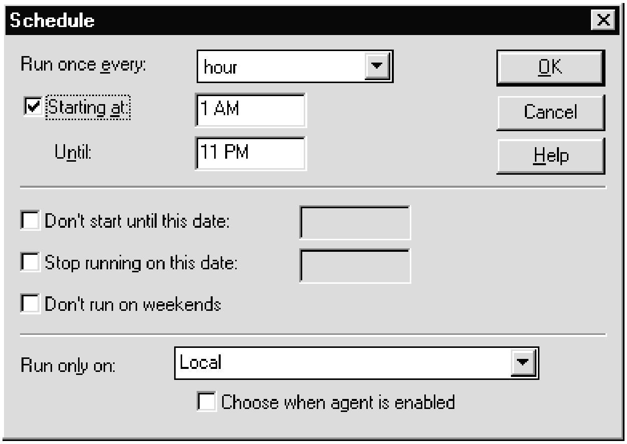
Caution Once you have saved an agent you cannot change a shared agent to a private agent or vice versa.

Scheduling the Agent

1. Next select when the agent will run. The following list of options is available:

   - Manually from Actions Menu
   - Manually from Agent List
   - If new mail has arrived
   - If documents have been created or modified
   - If documents have been pasted
   - Scheduled Hourly / Daily / Weekly / Monthly / Never

2. Select On Schedule Hourly from the pull-down menu and click the Schedule push button to schedule the specified run time. The following dialog box is displayed:



Schedule

Run once every: hour

☑ Starting at: 1 AM

Until: 11 PM

□ Don't start until this date:

□ Stop running on this date:

□ Don't run on weekends

Run only on: Local

□ Choose when agent is enabled

   The example shows an hourly scheduled macro. You can specify for the agent to run every 30 minutes, hour, every two, four, or eight hours.

3. Specify the start and end time each day.

4. Specify the start and end date for the agent to run, and whether to run at weekends or not.

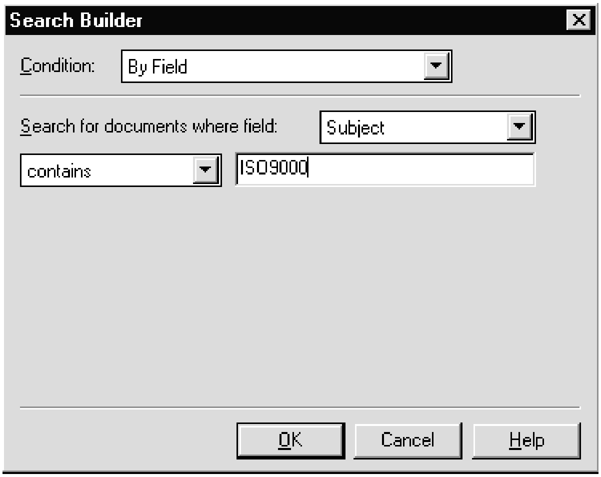
5. Specify the server on which the agent is to run. This is only applicable for databases which are replicated across a number of servers. If the agent is modifying data in the database, it should just run once on one server. Then the changed data is replicated to the other replicas of the database.

Selecting Documents to Be Processed

This selection defaults intelligently depending on the option selected for scheduling the agent. For example, if the agent is scheduled to run if new mail has arrived, this option is set to Newly Received Mail Documents, and it cannot be changed.

You can further select which documents are processed by specifying search criteria.

1. For example if you want to only process documents with ISO9000 in the subject, click on the Add Search push button to activate the Search Builder.



Search Builder

Condition: By Field

Search for documents where field: Subject

contains ISO9000

2. From the Condition drop-down box, select By Field.
3. In the Search for documents where field: box, select Subject.
4. Select Contains.
5. Type in the search criteria ISO9000.
6. Click on OK to save the search criteria.

Specifying What the Agent Should Do

There are three ways of specifying what the agent should do.

Simple Actions

These are pre-defined actions which allow you to define a sequence of actions without requiring any programming knowledge. They are ideal for the end user who wishes to automate some routine tasks. The simple actions available are:

Copy to database

Copy to folder

Delete from database

Mark document read

Mark document unread

Modify field

Modify fields by form

Move to folder

Remove from folder

Reply to sender

Run agent

Send document

Send mail message

Send newsletter summary

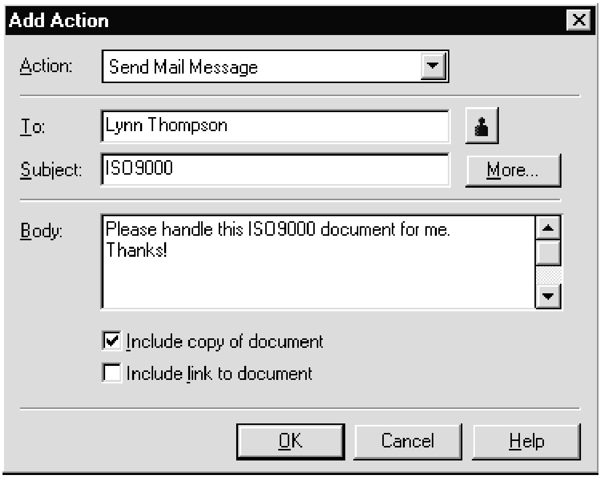
Run @Function formula

In our previous example, we used the Search Builder to select only those documents which had ISO9000 in the subject. Now we can easily write a simple action to automatically forward those mail items to the departmental ISO9000 coordinator and store the documents in the ISO9000 folder.

To do so:

1. Return to the Agent Builder window.
2. Click the Simple Action(s) option button.
3. Click the Add Action push button.
4. In the Action field, choose Send Mail Message.
5. In the To: field, type the name of the ISO9000 coordinator.
6. Type a subject.
7. In the Body: field, type your message.
8. Select the Include copy of document check box.

9. Click on OK. The completed Add Action dialog box looks like this:



Action: Send Mail Message

To: Lynn Thompson

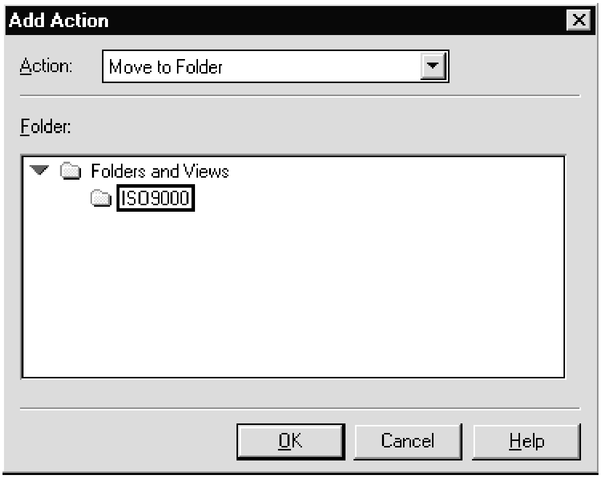
Subject: ISO9000

Body: Please handle this ISO9000 document for me.
Thanks!

☑ Include copy of document

□ Include link to document

10. Click on the Add Action push button again.
11. This time, select Move to Folder in the Action field:



Action: Move to Folder

Folder:

Folders and Views
ISO9000

12. Select the folder to move the document to, in our example ISO9000.
13. Click OK.

We now have an agent with two simple actions which will forward the document and then store it in a folder.

Formulas

Formulas can use the full range of @Functions available with Lotus Notes.

Note Interactive functions and functions that impede the progress of the mail router are ignored when documents are mailed into the database, for example:

- @DbColumn
- @DbCommand
- @DbLookup
- @MailSend
- @Prompt

1. In the Agent Builder window, click on the Formula option button.
2. Start entering a formula in the programmer pane. For example, if you want to forward a document but only if it does not have attachments:

```
@If( @Attachments>0;
   @Return("");
   @MailSend("Erika Smith"; ""; ""; Subject; _
             "Please handle this"+@Newline; "Body" ; "")
   );
```

LotusScript

Agents can also be written in LotusScript.

1. In the Agent Builder window, click on the Script option button.
2. Start entering a LotusScript program in the programmer pane.

Control is always passed to the agent using the Initialize event, so this is where the program should begin.

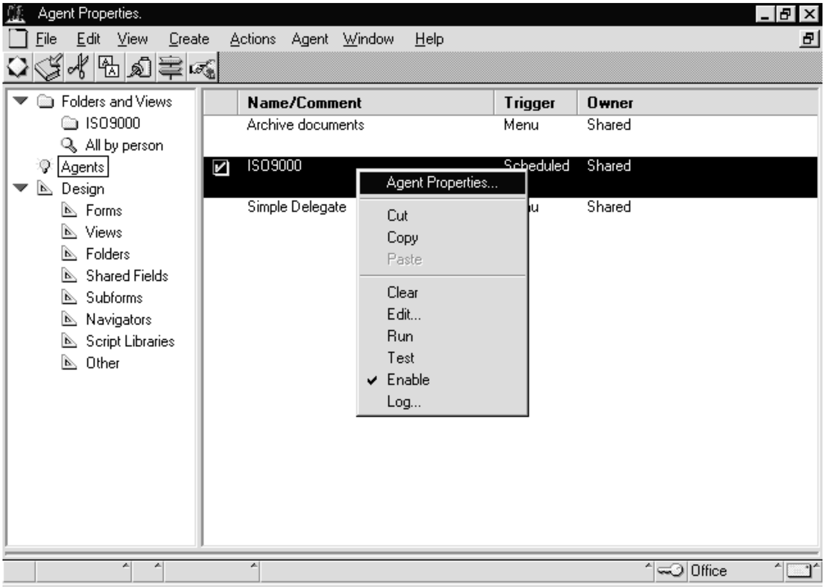
This is a simple example of a LotusScript program:

```
Sub Initialize
     Set s = New NotesSession
     Set db = s.CurrentDatabase
     Set documents = db.UnProcessedDocuments
     For d = 1 To documents.Count
          Set doc = documents.GetNthDocument(d)
          Set Subject = doc.GetFirstItem("Subject")
          If  Instr(Subject.Text , "ISO9000")>0 Then
               Call doc.PutInFolder( "ISO9000" )
          End If
     Next
End Sub
```

When an agent runs, it runs once and must process all the documents selected. Note the use of the UnprocessedDocuments property. The UnprocessedDocuments property of the NotesDatabase class applies only to agents and view actions. For agents, this property contains all documents not yet processed by the agent or the result of the search specified to the agent builder, depending on how you create the agent. For view actions, this property contains all selected documents.

Displaying the Agent Pop-up Menu

1. To display the pop-up menu of an agent, click with mouse button 2 on an agent listed in the navigation pane. The agent pop-up menu is displayed as follows:



	Name/Comment	Trigger	Owner
	Archive documents	Menu	Shared
☑	ISO9000	Scheduled	Shared
	Simple Delegate	u	Shared

2. From this menu you can:
   - Display the agent Info Box.
   - Cut and copy to the clipboard, and paste from the clipboard.
   - Clear the agent, which means deleting it.
   - Edit the agent, which is the same as double-clicking it, to display the Agent Builder window.
   - Run the agent.
   - Test the agent, which tells you how many documents the agent will process.
   - Enable or disable the agent.
   - Look at the agent log.

Summary

Agents can be used for automating many routine tasks in Notes. They have highly flexible scheduling capabilities, and a range of algorithms for selecting which documents should be processed.

They can be provided by developers as part of an application or installation, or can be set up by users to automate routine tasks such as filing or replying to mail.

They can be written in Notes Simple Actions which are ideal for the end user as no programming knowledge is required, or you can use the rich facilities of @Functions or LotusScript.

Chapter 9
Calendaring & Scheduling

Lotus Notes Release 4.5 now has native Calendaring & Scheduling capabilities built directly into the server and each client. This new feature allows users to create and manage day-to-day appointments in their mail file, book meetings with other people while checking their availability, look at two days, one week, two weeks or a month of their diary at a time and many additional features.

This chapter aims to focus on the new tools available to the Notes application developer from Calendaring & Scheduling, rather than a detailed description of how to use Calendaring & Scheduling in the new Release 4.5 mail file template.

This chapter was written using a beta copy of Notes Release 4.5 and as such, some of the information may have changed slightly in your shipped version of the product.

By the end of this chapter you will know:

- What Calendaring & Scheduling is.
- How to use the new LotusScript classes and methods related to Calendaring & Scheduling.
- How to use the new @Functions related to Calendaring & Scheduling.
- How to create a view using the new calendar view type.
- How to use the new date and time controls.

What Is Calendaring & Scheduling?

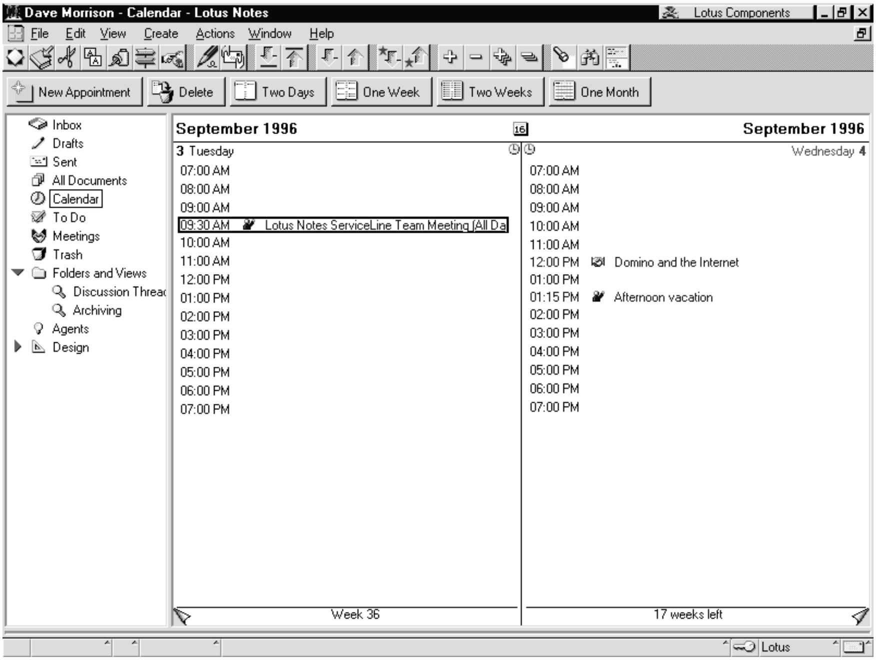
If you have ever used Lotus Organizer you will know that you can create appointments in your diary, create alarms to remind you of meetings you must attend and add your mother’s birthday so you don’t forget to send her a card!

Now, you can do all of these things using Lotus Notes and take it one step further. Not only can you use Notes as a diary and a reminder, you can now schedule meetings with other people, check their diaries to find the most convenient time to have the meeting, find a room in which to hold the meeting and reserve resources such as overhead projectors and video players.

Calendar Views

The Notes Release 4.5 mail database contains a new view called the Calendar view. This is a graphical representation of a diary, very much like that in Lotus Organizer, that you can use to show your appointments in two day, one week, two week or one month formats. Below are the four different calendar view types:

Two day view:



September 1996

3 Tuesday

07:00 AM
08:00 AM
09:00 AM
09:30 AM Lotus Notes ServiceLine Team Meeting (All Da
10:00 AM
11:00 AM
12:00 PM
01:00 PM
02:00 PM
03:00 PM
04:00 PM
05:00 PM
06:00 PM
07:00 PM

Week 36

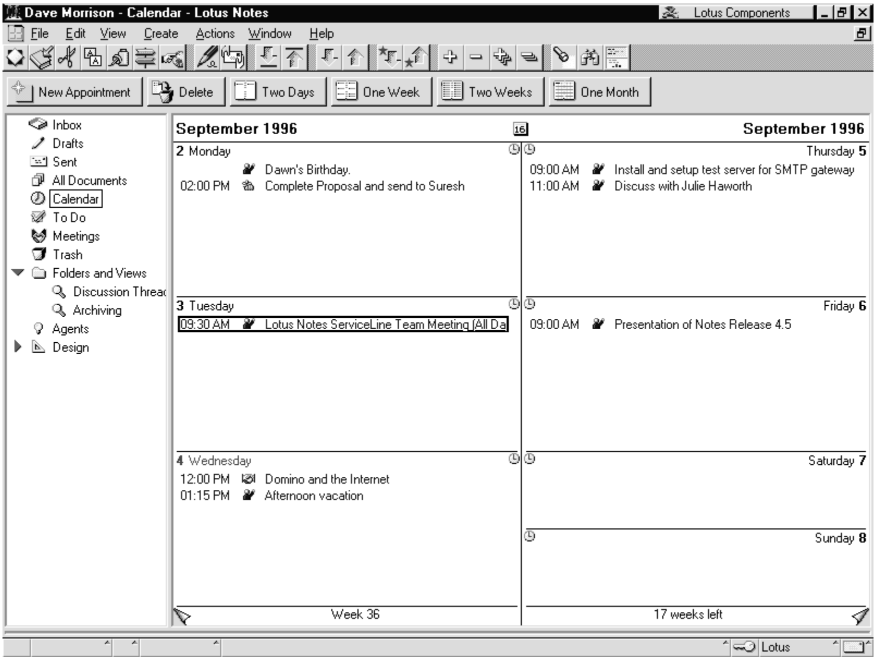
September 1996

Wednesday 4

07:00 AM
08:00 AM
09:00 AM
10:00 AM
11:00 AM
12:00 PM Domino and the Internet
01:00 PM
01:15 PM Afternoon vacation
02:00 PM
03:00 PM
04:00 PM
05:00 PM
06:00 PM
07:00 PM

17 weeks left

The week view:



September 1996 — September 1996

2 Monday
Dawn's Birthday.
02:00 PM Complete Proposal and send to Suresh

3 Tuesday
09:30 AM Lotus Notes ServiceLine Team Meeting (All Da

4 Wednesday
12:00 PM Domino and the Internet
01:15 PM Afternoon vacation

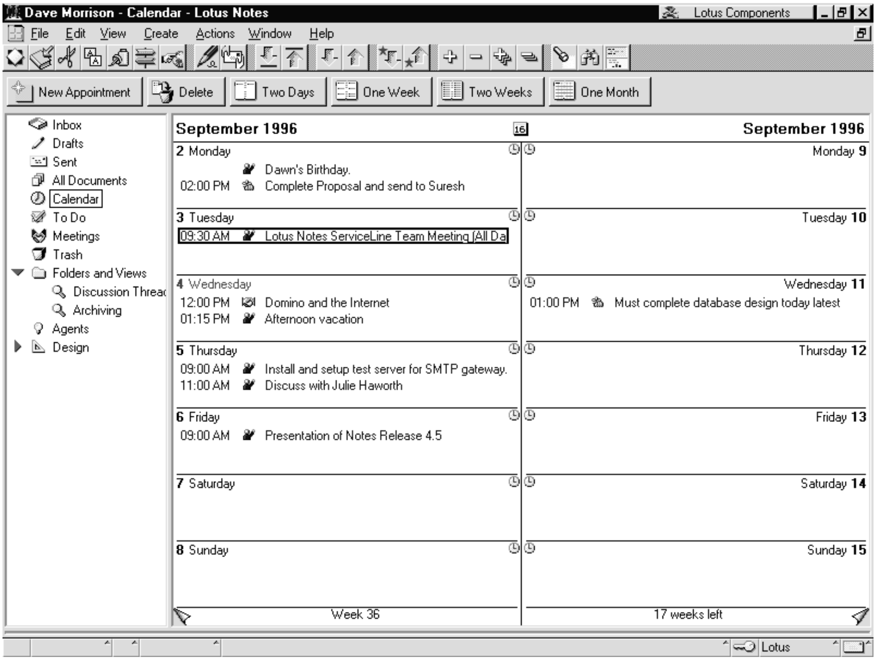
Thursday 5
09:00 AM Install and setup test server for SMTP gateway
11:00 AM Discuss with Julie Haworth

Friday 6
09:00 AM Presentation of Notes Release 4.5

Saturday 7

Sunday 8

The two week view:



September 1996 — September 1996

2 Monday
Dawn's Birthday.
02:00 PM Complete Proposal and send to Suresh

3 Tuesday
09:30 AM Lotus Notes ServiceLine Team Meeting (All Da

4 Wednesday
12:00 PM Domino and the Internet
01:15 PM Afternoon vacation

5 Thursday
09:00 AM Install and setup test server for SMTP gateway.
11:00 AM Discuss with Julie Haworth

6 Friday
09:00 AM Presentation of Notes Release 4.5

7 Saturday

8 Sunday

Monday 9

Tuesday 10

Wednesday 11
01:00 PM Must complete database design today latest

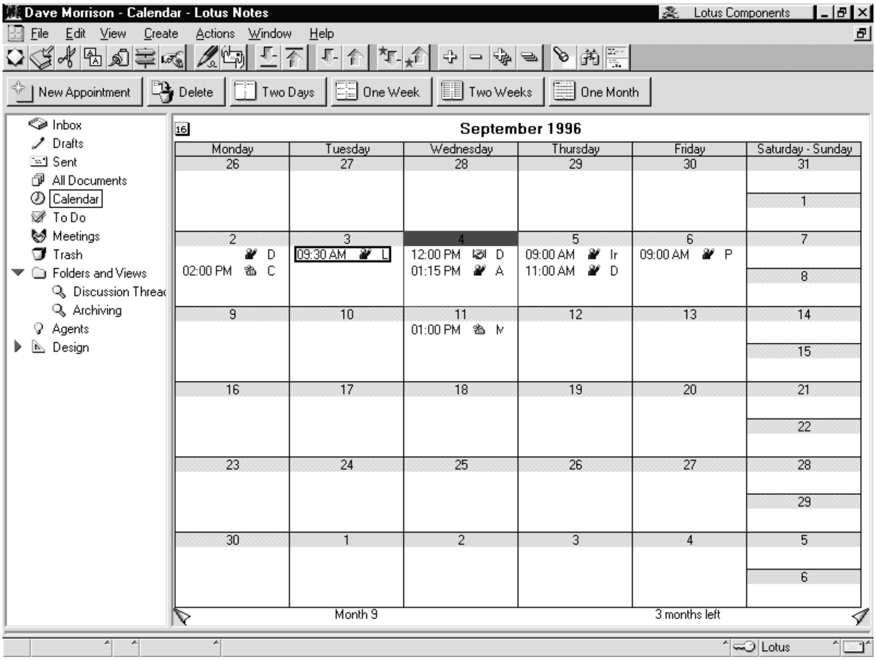
Thursday 12

Friday 13

Saturday 14

Sunday 15

The month view:



September 1996

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday - Sunday
26	27	28	29	30	31 1
2 D 02:00 PM C	3 09:30 AM L	4 12:00 PM D 01:15 PM A	5 09:00 AM Ir 11:00 AM D	6 09:00 AM P	7 8
9	10	11 01:00 PM M	12	13	14 15
16	17	18	19	20	21 22
23	24	25	26	27	28 29
30	1	2	3	4	5 6

Month 9 3 months left

To book an appointment in your diary you select the New Appointment action button which displays the New Appointment form. From here you can create a Personal Appointment, an Anniversary, a Meeting, a Personal Reminder or an Event.

Having selected the people you want to attend the meeting you can have Lotus Notes check their diaries to see if they are free on the date and time you specified.

To be able to check whether the invited people are free or not, Notes uses something called the Free-Time system.

Free-Time System

The Free-Time system is divided into three separate areas.

Checking for Users' Free Time

The Free-Time Manager is responsible for collecting all the available free time found in each user's mail file and posting it into the Free-Time database that is stored on the server.

Free-Time Database

The second element is the Free-Time database. When the Free-Time Schedule manager starts up for the first time on the server, it creates a new database called BUSYTIME.NSF and populates it with the names of all the people found in the Public Name and Address Book that have their mail server set as the current server's name.

The Free-Time database contains the following fields.

Field Name	*Field Description*
$BusyName	The distinguished name of the person / resource / room that will be busy for the period.
StartDateTime	A Notes DateTime field specifying the beginning of the busy period.
EndDateTime	A Notes DateTime field specifying the end of the busy period.
BusyPriority	A number between 1 (lowest, pencilled in) to 15 (highest). 15 means that the person is not available, that is, out of the office, on vacation and so on.
BusyType	A number between 1 and 15. A value of 0 (zero) means no type.
WorkWeek	From the user's profile document. A list of DateTime ranges that shows the allowable free times in the week. Each entry corresponds to a day in the week.
ScheduleAccess	From the user's profile document. A list of people who can create busy time for this user.
TimeZone	From the user's profile document. The time zone of the user.
$BusyProfile	A marker that tells the Free-Time Database Manager that this record contains user-specific Free-Time database information.

This database can only be accessed by the Free-Time manager and is structured to allow fast querying of people's schedules.

Calendar Connector

Lastly, there's the Calendar Connector. When a client asks the free-time manager whether a person is free to attend a meeting it will do a number of tasks.

1. Look the person up in the Name and Address Book.
2. If the person's free-time information is stored on that particular server it will look in the free-time database.
3. If the user is found to be on another server, the request is passed to the Calendar Connector server task. If the person is in the same domain, the Calendar Connector will chain the request to the appropriate server.

4. If the person is in an adjacent domain, the Calendar Connector uses the information stored in the Calendar Domain field in the corresponding domain document in the Public Name and Address Book. Each domain has a designated calendar server that is set up to receive requests from external domains. When a request is sent from one domain to another, the designated calendar server is responsible for receiving the query, passing it to the correct server within its own domain and returning the information back to the requesting domain.

5. If the person is in a non-Notes foreign domain, the Calendar Connector uses two values from field information stored in the foreign domain document in the Public Name and Address Book. The first is the Calendar Server Name and the second is the Calendar System. This second field stores the type of calendaring system that the calendar connector should connect to, such as Organizer 2.1 or IBM OfficeVision, and then makes the request to that system through a plug-in.

   Note Lotus will develop calendar plug-ins for Lotus Organizer 2.1 and for OfficeVision. Lotus will also develop tools to simplify the migration to Notes 4.5 from either of these systems.

6. The returning result of the query is then passed back to the client.

Resources

Notes Release 4.0 included a Room Reservation template, RESERVE4.NTF, that allowed you to create resources and have people book them. This has been enhanced in Notes Release 4.5 to enable Calendaring & Scheduling users to book meeting rooms and resources and to interact with the Free-Time system.

Resources are split into two separate area's, *Rooms*, that are physical meeting places within a location that can contain up to a maximum number of specified people and *Other Resources* that include items such as overhead projectors, video players, televisions etc. that can be booked together with a room.

Once a reservation database has been created, designated *Resource Creators* can add sites, rooms and resources to the database for other people to use. When a resource is created within the database, a corresponding item is added to the Public Name and Address Book in the Mail-In Databases and Resources view.

Essentially, a resource database is an extension to the normal mail-in database that is integrated into the scheduling system.

Programming With Calendaring & Scheduling

Lotus Notes Release 4.5 has a number of new LotusScript commands and @Functions to support Calendaring & Scheduling.

LotusScript

Below is a summary of the LotusScript properties, events and methods.

Properties

`NotesDocumentCollection = NotesUIView.Documents`

Contains the documents that the current event will act on.

`Boolean = NotesUIView.IsCalendarView`

True if the current view is the calendar style rather than the traditional style.

`NotesDateTime = NotesUIView.CalendarDateTime`

This is only valid for calendar views and will allow you to set or query the current date and time for the view.

Events

`NotesUIView.QueryOpenDocument`

Allows you to query if a document is being opened.

`NotesUIView.RegionDoubleClick`

Triggered when a user clicks on a region within the Calendar view that is defined by the type of view currently open. There are two regions in the Calendar view, the specific time slots and the blank area within a day.

`NotesUIView.QueryPaste / NotesUIView.Paste`

Allows you to handle paste commands from the user.

`NotesUIView.QueryDragDrop / NotesUIView.Drop`

Allows you to handle drag and drop operations from the user.

Note You should first set the value of CalendarDateTime to the date and time you wish to paste or drop the documents into.

`NotesUIView.QueryDelete / NotesUIView.Delete`

Allows you to handle delete requests from the user.

Methods

```
Set NotesDateRange =
notesSession.FreeTimeSearch(window,duration,names
[,firstfit])
```

Creates a call to the Notes Free-Time Manager requesting the availability of one or more people, where *window* is a NotesDateRange for the start and end dates in which the search is to be performed, *duration* is an integer in minutes and *names* is a string or an array of strings containing the names of the people for the request to be performed on.

@Functions

Below is a summary of the @functions available for Calendaring & Scheduling.

@Function name	*Description*
@CheckAlarms	Triggers the alarm daemon to check for new alarms in the mail file.
@EnableAlarms(0 or 1)	Starts or stops the alarm daemon.
@GetProfileField(profilename;fieldname [;username])	Retrieves a field from a profile document, and caches the field value for the remainder of the session.
@SetProfileField(profilename;fieldname; value [;username])	Sets the value of a field in a profile document.
@Command([CalendarFormat];*number*)	Changes the calendar view to a different format where *number* is 2, 7, 14, 30.
@Command([EditProfile]; "profilename"; "username")	Opens a new or existing profile document in edit mode.
@Command([CalendarGoto];*timedate*)	Goes to a particular date in a calendar view.
@Command([FindFreeTimeDialog];[*req people*]; [*optpeople*]; [*rooms*]; [*optrooms*]; [*reqresource*]; [*optresource*]; [*removed*]; [*StartDateTime*]; [*EndDateTime*])	Opens the Free Time dialog box to allow searches for available meeting times. Optional parameters allow the developer greater flexibility.

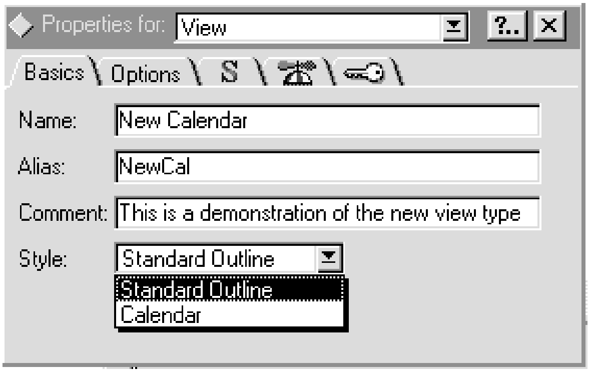
The Calendar View

From a designer's perspective, the calendar view is identical to the standard outline view, with columns, formulas, a selection criteria and fonts.

To create a calendar view there are a number of steps you must follow.

1. Firstly, create a new view in the database you wish to add the calendar view to by selecting Create-Design-View from the Notes menu bar. Open the new view by double-clicking on its name.

   Select View-Design Properties from the menu to display the View InfoBox. Click on the style drop-down listbox and select the Calendar style.



Properties for: View

Name: New Calendar

Alias: NewCal

Comment: This is a demonstration of the new view type

Style: Standard Outline

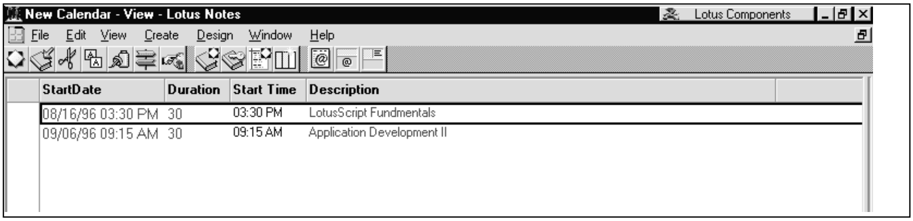
2. For documents to be displayed in a calendar view you need to provide a date, a time and a description of the event. At a minimum you must provide the view with a start date and a textual description. To create the view you must set up the columns in the correct order.

   - The first column must contain a valid date value and it must be sorted. To improve the view, add a time value to the date. This column should be hidden.
   - The second column should contain the duration of the event in minutes or zero for no duration. This field should be hidden.
   - The third and subsequent columns contain the text that is displayed for the event.
   - For time-based events, you can use the third column to hold the starting time of the event and the fourth column to hold the description. This ensures that the event appears in the correct time slot.

Tip To have an event displayed in a calendar view with Enable Time Slots enabled in the View InfoBox on the date format tab, set the third column to contain the start time, change the date format to display only hours and minutes and change the font size of the column to Helv 8 point.

3. You must also create a view selection statement so that these columns are populated correctly.

This is how the view looks when in the design pane when you have added the columns.



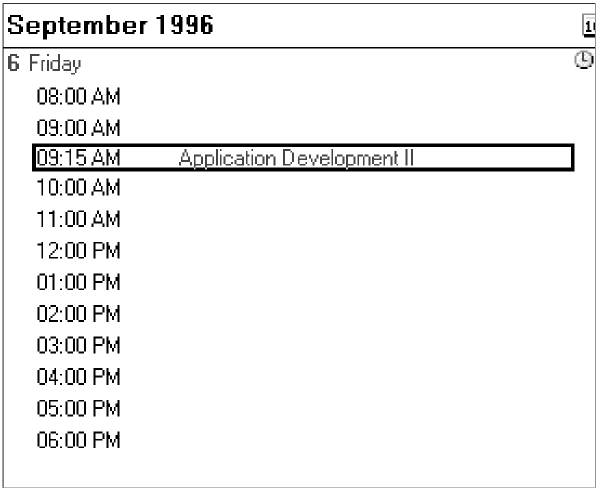
StartDate	Duration	Start Time	Description
08/16/96 03:30 PM	30	03:30 PM	LotusScript Fundmentals
09/06/96 09:15 AM	30	09:15 AM	Application Development II

These are the formula's for each of the columns:

StartDate is a date-time variable that contains only a date, StartTime is a datetime variable that only contains a time. This example shows how to join the two together to create a single date-time field.

Column	*Formula*	*Attributes*
1	@TextToTime(@Text(StartDate) + "" + @Text(StartTime))	Hidden, Sorted
2	30	Hidden
3	StartTime	Formatted for hours and minutes only, blue Helv 8 font.
4	SeminarName	Helv 8 point font.

This is how the item appears when displayed using the calendar view.



September 1996

6 Friday

```
08:00 AM
09:00 AM
09:15 AM        Application Development II
10:00 AM
11:00 AM
12:00 PM
01:00 PM
02:00 PM
03:00 PM
04:00 PM
05:00 PM
06:00 PM
```

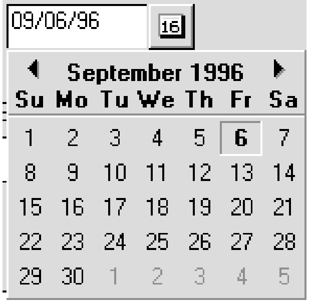
Note It is possible to create a calendar view with just a date (no time) and description column, however, you will not be able to use the Enable Time Slots feature.

Date and Time Controls

It is now possible from Notes Release 4.5 to include pop-up controls for date and time fields in a layout region that is designed to behave in the same way as the Lotus Organizer controls.

Date Control

The date control allows the user to quickly select a date by clicking on the small calendar icon next to the field in which it relates to. When the user clicks on the icon, a pop-up box will appear in which they can scroll forwards and backwards though the months of the year and select an appropriate date by clicking on it with the mouse.



09/06/96

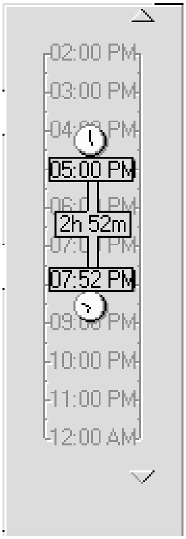
September 1996

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Note To create the date pop-up control, the field must be in a layout region, it must be set to a time type and set to show a date format.

Time Control

The time control is a pop-up box that enables the user to select either a single time or a duration by using the mouse to drag the time bar up and down the scale.



To create a simple time control, create a field in a layout region and assign the type to time and to show only a time format.

Note To add the duration to the pop-up, select the Allow Multi-Values check box on the InfoBox for the field.

Chapter 10
Notes Workflow: An Example

Creating a Database Using the Approval Cycle Template

Notes Release 4 provides many database templates that can be used to create Notes databases. Some of the templates are new in Release 4.

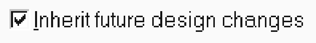
You can use the templates both for typical office work and for Notes database administration. For example, to look at the templates you can use for administration purposes:

1. Choose File - Database - New.
2. Check the Show advanced templates check box:

   ☑ Show advanced templates

   Additional templates are displayed.

You can use the templates to create simple databases without modifying the template. In this case, you would typically check the Inherit future design changes check box on the New Database dialog box. This ensures that changes made to the original template will be reflected in your version of the database.



☑ Inherit future design changes

Or, you can use the templates to create customized databases and modify the template as required. In this case, you would typically deselect the Inherit future design changes check box on the New Database dialog box. Changes made to the original template will not be reflected in your version of the database.

□ Inherit future design changes

The Approval Cycle Template

In this chapter, you are going to work with a workflow database which is based on the Approval Cycle template. This template is suitable for workflow types of applications, for example, Asset Purchase Request, Travel Expense Account Approval, and the like.

The Approval Cycle template provides design forms and a design subform.

Note The source code of the Approval Cycle template is provided in Appendix A.

You are going to look at the Application Profile form and the Approval Logic subform. You can use the Application Profile form to describe the characteristics of a workflow object, such as a Travel Expense Account. You can use the Approval Logic subform for an application request form. You can define the required parameters in the Approval Logic subform, such as how many people will approve the travel expense, by what date, and status-specific information.

This is the specification of the Application Profile form:

- Routing Type: Serial or Parallel

  Serial routing means that an approval request is sent to another approver once the current approver has finished working with it. Parallel routing means that an approval request is sent to all approvers at the same time.

- Routing Delivery Type: Document Link

  Document Link is the default routing delivery type. Approvers open their mail and see a document link icon pointing to a specific approval request sent by another user. When they click the link icon, the approval request is opened.

  If the routing delivery type is Entire Document, approvers receive the complete document in their mail, not a link to the document.

- Number of Approvers: 1 to 5

  The design of the Approval Cycle template provides for a number of approvers between one and five. However, there is no limit on the number of approvers you can assign. Keep in mind, though, that there is a push button created for each approver. If you increase the number of approvers, there will be a corresponding number of push buttons, which can easily fill up the screen.

- Method to specify an approver: Define in Profile, Entered on the form by the submitter, Retrieved from a database.

  If you choose Define in Profile, you need to specify an approver when you create an application profile.

  If you choose Entered on the form by the submitter, the submitter of the request form needs to select the approvers when creating an approval request.

  If you choose Retrieved from a database, you need to specify an Address Book name in a following dialog box. Approvers are selected based on the Manager field of an originator's document in the Address Book you specified.
- Actions to be taken: Approve, Deny.
- Upper limit for each part of the approval cycle: Number of days.

  You can specify how many days an approver can take to complete the approval request.
- Action for expiration: Approve the form, Reject the form, Send a reminder, Do nothing.

  When the specified number of days has passed for each part of the approval cycle, one of the above-listed actions is performed.

Using the Approval Cycle Template

You usually need to prepare an application form in advance to use the Approval Cycle template. In our example, we have written a form that contains a Travel Expense Account (TEA) workflow, and have added it to the Approval Cycle template.

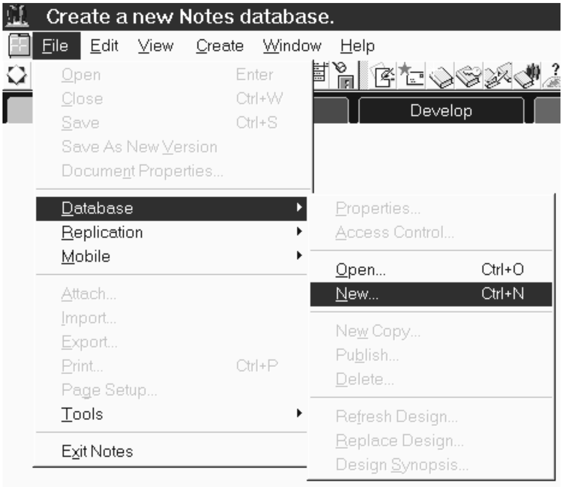
The TEA form consists of two subforms:

- The TEA subform, which we added.
- The Approval Logic subform, which is included in the Approval Cycle template.

Adding the Database to Your Workspace

To add the Approval Cycle template to your workspace:

1. Choose File - Database - New:



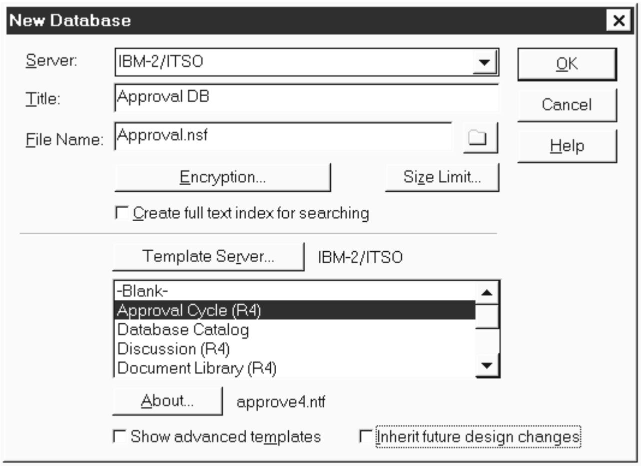
Tip A shortcut is pressing Ctrl - N.

2. Type a server name, database title, and database file name. In our example, we called the database Approval DB.

3. Click the Template Server push button to select a server for the template.

4. Choose the Approval Cycle template from the list of templates.

5. Deselect the Inherit future design changes checkbox. The completed New Database dialog box looks like this:



New Database

Server: IBM-2/ITSO

Title: Approval DB

File Name: Approval.nsf

□ Create full text index for searching

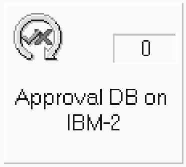
Template Server... IBM-2/ITSO

-Blank-
Approval Cycle (R4)
Database Catalog
Discussion (R4)
Document Library (R4)

About... approve4.ntf

□ Show advanced templates □ Inherit future design changes

6. Click OK.
7. Press the **ESC** key to leave the information window. The Design toolbox is shown.
8. Press the **ESC** key to leave the Design toolbox. Your version of the Approval database has been added to your Notes workspace.



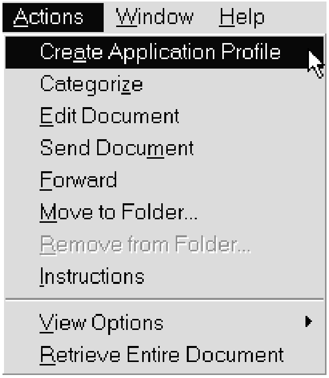
Note You also need to specify the appropriate Access Control List (ACL) for this workflow example. In our example, we have defined four users: Form Administrator, Request Originator, Manager, Accountant. All of them have access to the Approval database.

Caution Users accessing the Approval Cycle database must have access to the Notes Release 4 Address Book. Some functions do not work with a Release 3 Address Book.

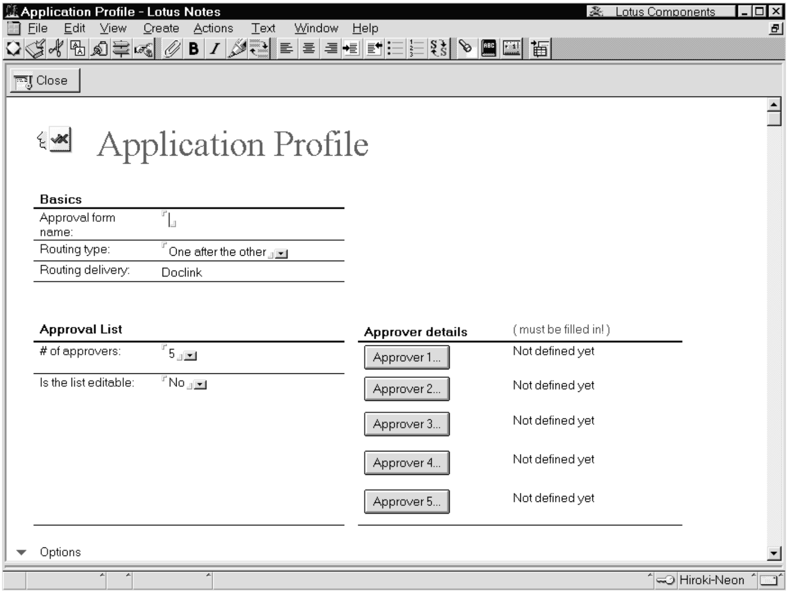
Creating an Application Profile

First create an application profile for the application form.

1. Open the Approval DB database.
2. Choose Actions - Create Application Profile from the menu:



3. The Application Profile is displayed. It looks like this:


Application Profile - Lotus Notes
Lotus Components
File Edit View Create Actions Text Window Help
Close
Options
Hiroki-Neon

Application Profile

Basics

Approval form name:	
Routing type:	One after the other
Routing delivery:	Doclink

Approval List

# of approvers:	5
Is the list editable:	No

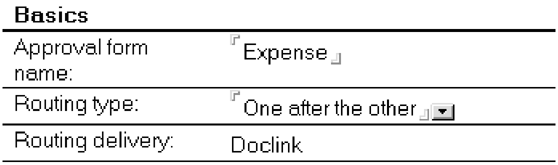
Approver details (must be filled in!)

Approver 1...	Not defined yet
Approver 2...	Not defined yet
Approver 3...	Not defined yet
Approver 4...	Not defined yet
Approver 5...	Not defined yet

4. In the Basics section, specify a name for the approval form.

   Note This is a required field. The name you specify here must be the same as the Application Form name you will define later on.

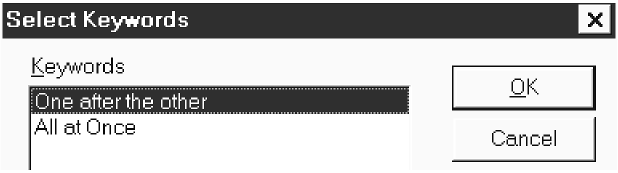
5. Click the arrow next to Routing type to display the available options:



Basics

Approval form name:	Expense
Routing type:	One after the other
Routing delivery:	Doclink

These are the available Routing type options:

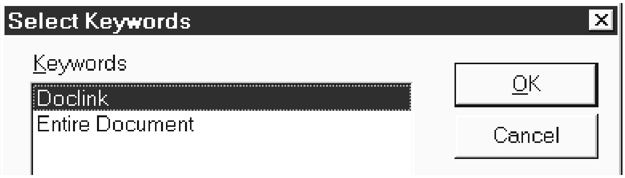


Keywords

One after the other
All at Once

6. Click OK. Leave the Routing type value unchanged.

7. Click the arrow next to Routing delivery to display the available options:



Note If there is no arrow available next to Routing delivery, follow these steps to create one:

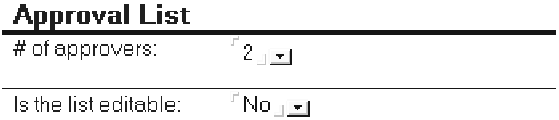
- Return to the Design toolbox window.
- Click Design, then Forms.
- Open the Application Profile listed in the view pane.
- Position the cursor in the Routing delivery field and click mouse button 2 to display the Field InfoBox.
- Change the value displayed in the Type field from Computed to Editable.
- Close the InfoBox.
- Close and save the form.

8. Click OK. Leave the Routing delivery value unchanged.

9. In the Approval List section, specify the number of approvers. The range is 1 to 5. In our example, we specified 2 approvers (the supervisor and the accountant).

Specify whether or not the approver list is editable. In our example, it is not editable.

The Approval List section looks like this:

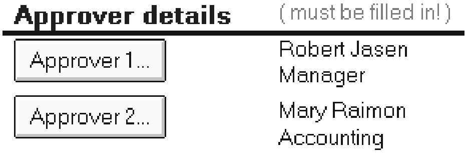


Approval List

# of approvers: 2

Is the list editable: No

10. In the Approver details section, you see two buttons, one for each approver. They look like this:



Approver details (must be filled in!)

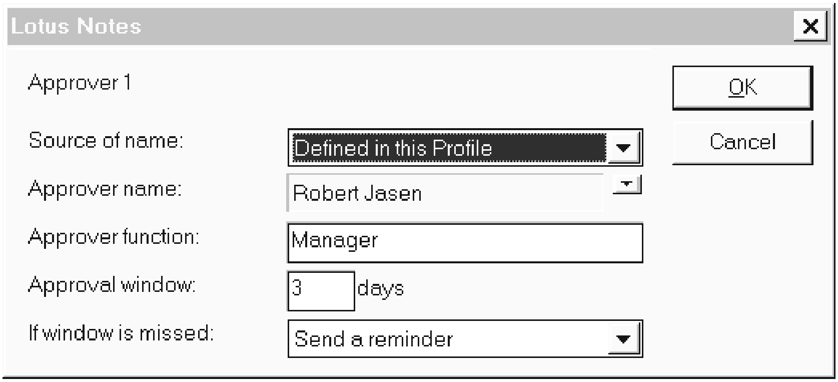
Approver 1... Robert Jasen
Manager

Approver 2... Mary Raimon
Accounting

11. Click the Approver 1 button to display a dialog box where you can specify information related to the approver.

    Note If this does not display the dialog box, click the button again.

    The dialog box looks like this:



Approver 1

Source of name: Defined in this Profile

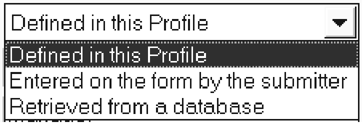
Approver name: Robert Jasen

Approver function: Manager

Approval window: 3 days

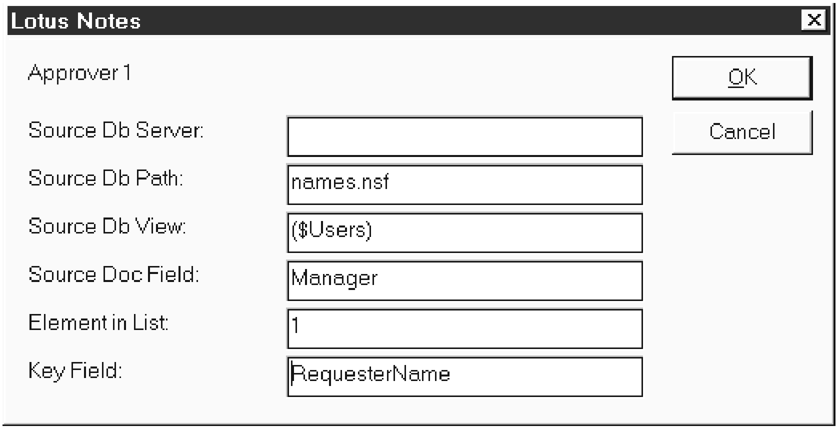
If window is missed: Send a reminder

12. Click the arrow next to Source of name to display the available choices:



Defined in this Profile
- Defined in this Profile
- Entered on the form by the submitter
- Retrieved from a database

13. If you select the Retrieved from a database option and click OK, a dialog box is displayed. You need to specify the following information:



Approver 1

Source Db Server:

Source Db Path: names.nsf

Source Db View: ($Users)

Source Doc Field: Manager

Element in List: 1

Key Field: RequesterName

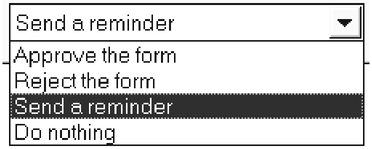
In our example, we left the Source of name field unchanged. This will make it a bit easier to perform our workflow example.

14. Click the arrow next to Approver name to display the Address book from which to select the approver name.

15. Specify the appropriate title for the approver in the Approver function field.

16. In the Approval window field, specify how long the approver can take to handle the approval request.

17. In the If Window is missed field, specify the action to be taken when the approval window has passed. These are the available options:



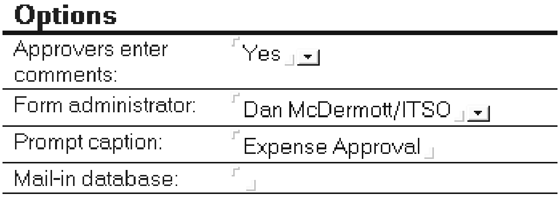
In our example, we selected Send a reminder.

18. Click OK to return to the Application Profile form.

19. Click the Approver 2 button to specify the required information for the second approver.

20. Optionally, fill in the Options section. In our example, we specified Yes for Approvers to enter comments when they approve or reject a request.

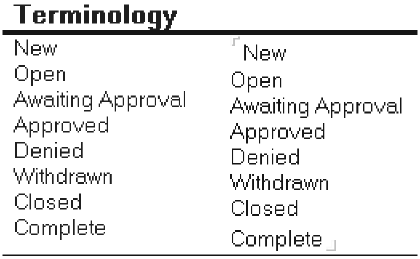
21. We also assigned a form administrator.



Options

Approvers enter comments:	Yes
Form administrator:	Dan McDermott/ITSO
Prompt caption:	Expense Approval
Mail-in database:	

22. You would not usually modify the Terminology section shown at the end of the Application Profile form. It shows information related to the workflow status. This information is displayed in an application request form to indicate the current status.



Terminology

New	New
Open	Open
Awaiting Approval	Awaiting Approval
Approved	Approved
Denied	Denied
Withdrawn	Withdrawn
Closed	Closed
Complete	Complete

23. Click the Close button and save the new document.

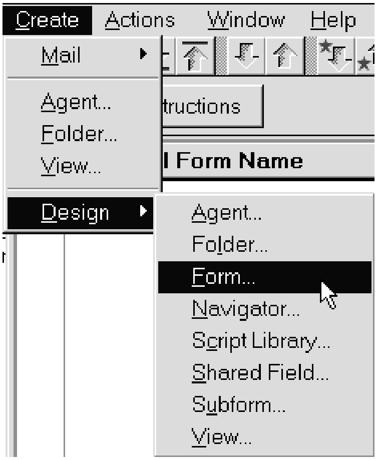
Creating the Approval Request Form

In the following, you are going to create a Travel Expense Account (TEA) approval request, which will be sent to two approvers. You will learn how to use the Approval Logic subform.

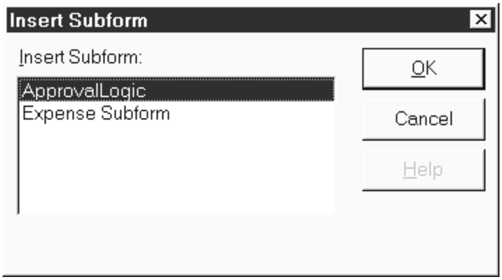
This example is of type document link, which means that all the approvers share the same approval request. You can also create a non-share type of workflow which means that an entire approval request form is sent to the approvers' mail box. If you want this type of workflow, you need to select the Entire document option as Routing delivery type, as explained in the previous section. However, selecting this type of workflow would require a more complicated approval request form than we use in our example here.

To create the approval request form:

1. On the Design toolbox, choose Create - Design - Form from the menu.

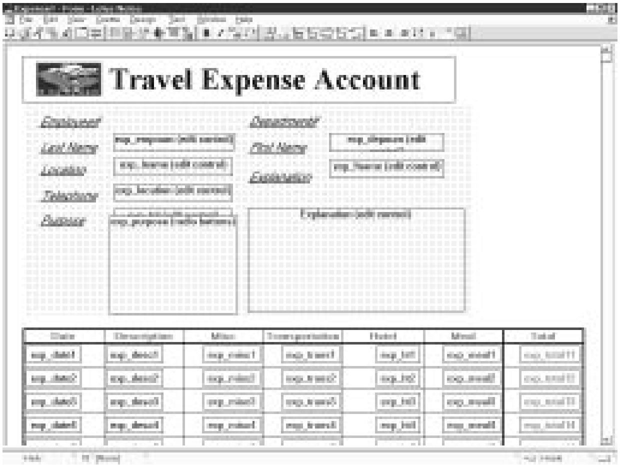


2. The Insert Subform dialog box is shown. Select Expense Subform.



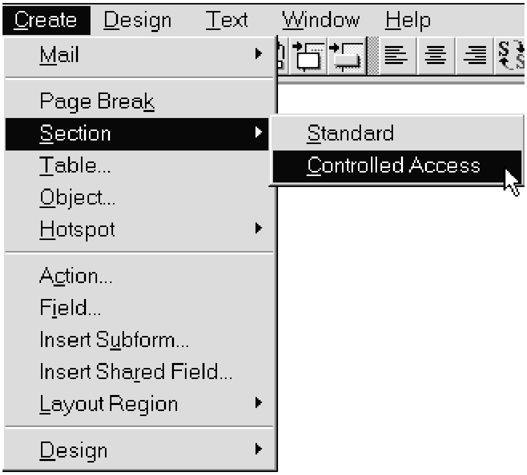
Note We created the expense subform. It is not provided as part of the Approval Cycle template. You need to create your own subform or form as required. For details on how to do this, refer to the chapter on Designing Application Forms.

3. Click OK. Our example of the expense subform looks like this:



4. Move to the end of the subform where you can see the blinking cursor. This is where you are going to create a collapsed section.

5. Choose Create - Section - Standard.



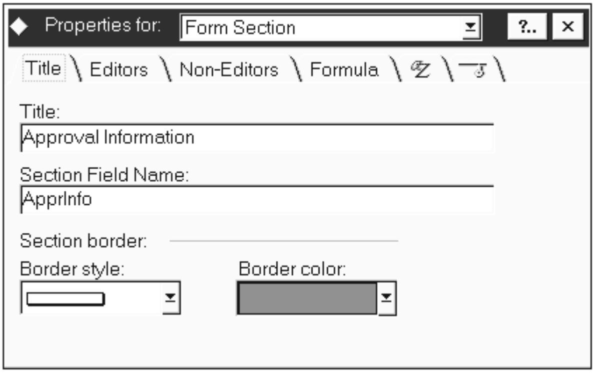
6. On the InfoBox, type the collapsed section title. In our example, we typed Approval Information.

7. Type the section field name. In our example, we typed ApprInfo.

8. In the Section border area, we chose a border style with an outline.

   Tip It is quite easy to insert a subform or other data into a collapsed section if you choose a border style with an outline.

The completed InfoBox looks like this:



Properties for: Form Section

Title: Approval Information

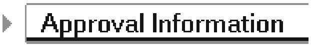
Section Field Name: ApprInfo

Section border:

Border style:

Border color:

Notice how the newly created section appears at the cursor position. This is what it looks like:

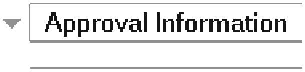


9. Close the InfoBox.
10. Click on the title of the new collapsed section to expand it.

    Tip You can also click on the triangle next to the title.

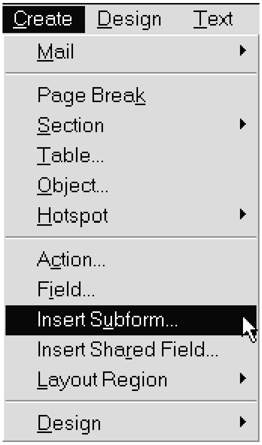
11. Before you can insert a subform, you need to follow these steps:

    Place the cursor on the bottom line of the box surrounding the Approval Information section, and click once. You should now see this:

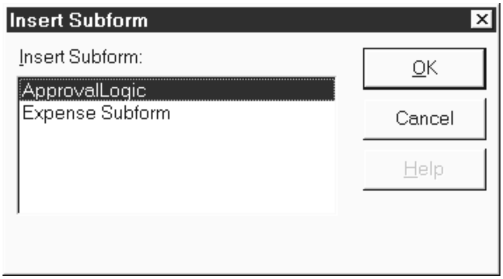


12. Click on the line displayed under the Approval Information section. The cursor should now be blinking on the new line.

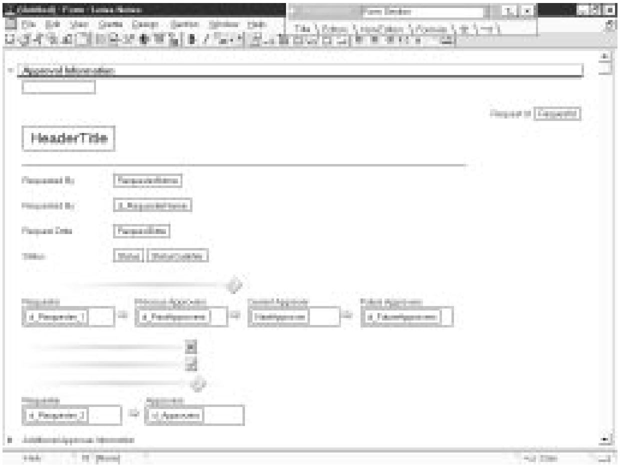
13. Choose Create - Insert Subform from the menu.



14. On the Insert Subform dialog box, select the ApprovalLogic subform.



15. Click OK. The design of the ApprovalLogic subform is displayed, as shown in the following figure:



Click on the Approval Information title. The section is collapsed.

16. Click mouse button two to display a pop-up menu and select Form Properties to display the InfoBox.

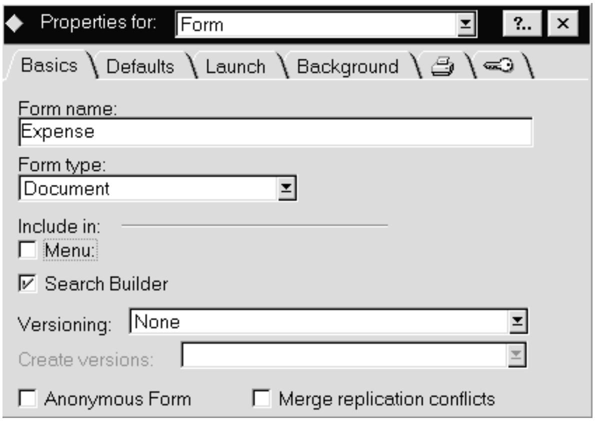
    Tip A shortcut for displaying the InfoBox box is pressing **ALT** and **ENTER**.

17. Type the form name.

    Note The form name must be the same as the name specified for the application profile you created before.

18. In our example, we deselected Include in Menu.

    The completed InfoBox looks like this:



Properties for: Form

Basics | Defaults | Launch | Background

Form name: Expense

Form type: Document

Include in:

□ Menu:

☑ Search Builder

Versioning: None

Create versions:

□ Anonymous Form

□ Merge replication conflicts

19. Choose Close and save the new form.

Performing a Workflow

In our example, four users are involved in the workflow:

Dan McDermott, form administrator

Nick Clark, submitter of a request

Robert Jasen, manager of Nick Clark

Martha O’Reilly, accountant.

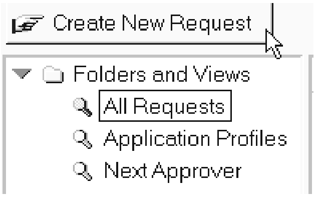
All the users have access to the Approval database. Make sure that you give your users access to the database as required.

Creating a New Request

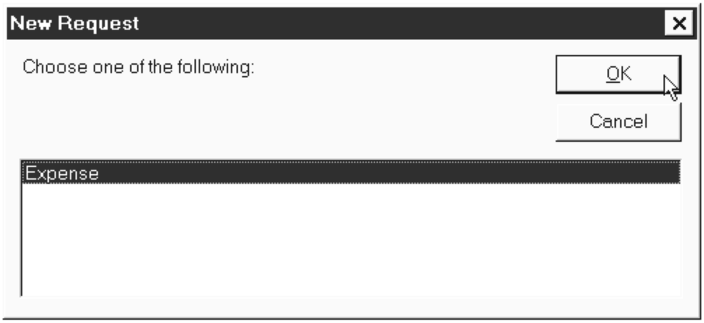
You are now Nick Clark, the submitter of a request. To create the request, follow these steps:

1. The Approval DB database should be open.

   From the navigator pane, choose All Requests under Folders and Views:

2. Click the Create New Request button. The following New Request dialog box is shown:

   Choose one of the following:

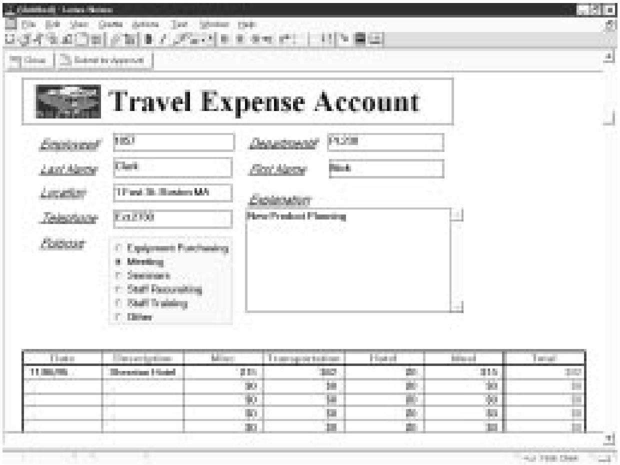
   Expense

   All the request forms that you created before are listed in the box.

   Note The form names are taken from the Workflow Object column that you see when you select Application Profiles in the navigator pane.

3. Choose the Expense form, which is our approval request form, and click OK.

4. Fill in the fields as required, for example:



Travel Expense Account

Last Name: Clark

First Name: Nick

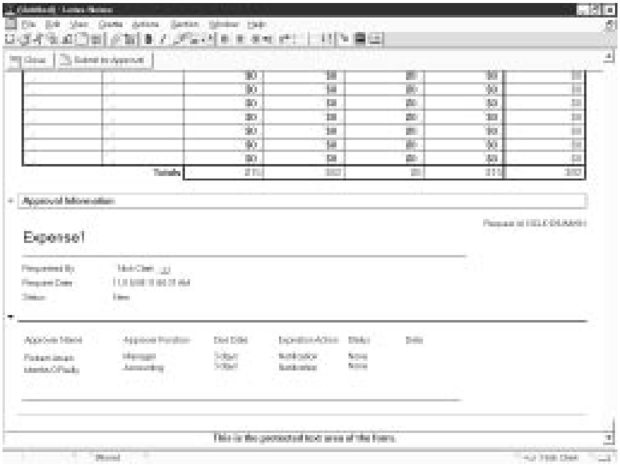
Explanation: New Product Planning

Purpose:
○ Equipment Purchasing
● Meeting
○ Seminars
○ Staff Recruiting
○ Staff Training
○ Other

5. Move to the bottom of the request form.

6. If the Approval Information section is collapsed, click on its title to expand it.

7. If the Additional Approver Information section is collapsed, click on its title to expand it.



Totals

Approval Information

Expense!

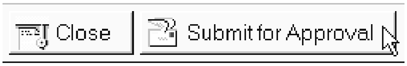
Status: New

This is the protected text area of the form.

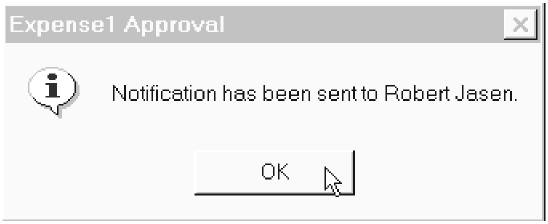
Submitting the Request

To submit the request:

1. Click the Submit for Approval button.



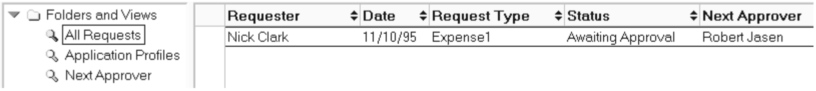
The following message is displayed:



Expense1 Approval

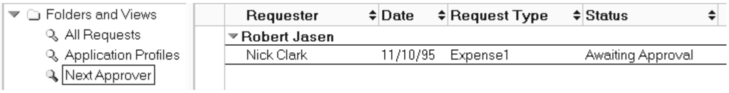
Notification has been sent to Robert Jasen.

2. Click OK to leave the message window. You can see the new request listed in the Date column. The status is Awaiting Approval. It looks like this:



Requester	Date	Request Type	Status	Next Approver
Nick Clark	11/10/95	Expense1	Awaiting Approval	Robert Jasen

3. Select Next Approver in the navigator pane. The view on the right-hand side now looks like this:



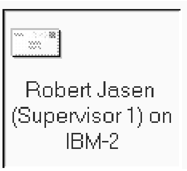
Requester	Date	Request Type	Status
Robert Jasen			
Nick Clark	11/10/95	Expense1	Awaiting Approval

4. Close the database.

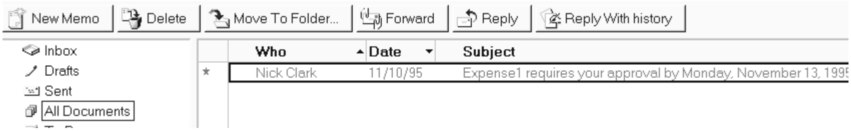
Working with the Request: The Manager

You are now Robert Jason, the manager.

1. Open your mailbox.



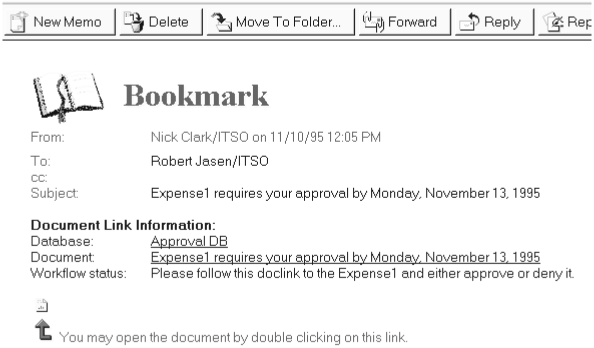
You can see mail listed in the All documents view. The mail has been sent by Nick Clark:



Who	Date	Subject
Nick Clark	11/10/95	Expense1 requires your approval by Monday, November 13, 1995

2. Open the mail sent by Nick Clark.

3. To open the approval request, double-click the document link icon pointed to by a red arrow:



Bookmark

From: Nick Clark/ITSO on 11/10/95 12:05 PM

To: Robert Jasen/ITSO

cc:

Subject: Expense1 requires your approval by Monday, November 13, 1995

Document Link Information:

Database: Approval DB

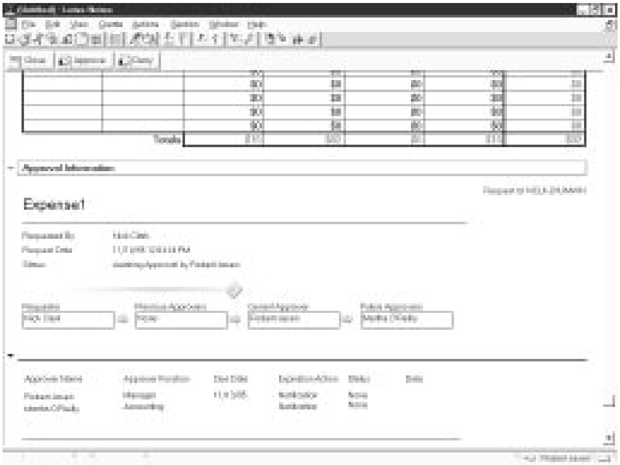
Document: Expense1 requires your approval by Monday, November 13, 1995

Workflow status: Please follow this doclink to the Expense1 and either approve or deny it.

You may open the document by double clicking on this link.

4. Move to the bottom of the document.

5. To see the due date and other workflow-related information, expand the Approval Information and Additional Approver Information sections. This is what you can see:



Expense1

Requested By: Nick Clark

Request Date: [illegible]

Status: Awaiting Approval by Robert Jasen

Requester: Nick Clark

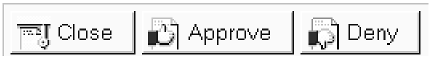
Previous Approvers: None

Current Approver: Robert Jasen

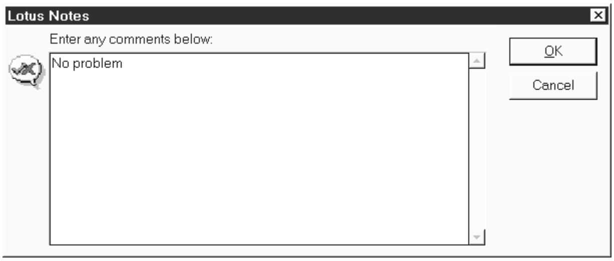
Future Approvers: Martha O'Reilly

Approver Name	Approver Position	Due Date	Expiration Action	Status	Date
Robert Jasen	Manager	11/13/95	Notification	None	
Martha O'Reilly	Accounting		Notification	None	

The action bar at the top of the window shows three buttons:



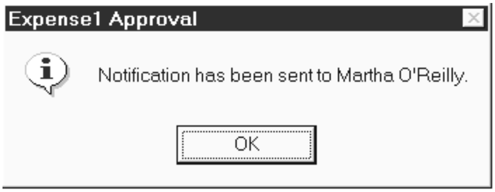
6. Click the Approve button to approve this request.
7. Type your comments as required.


Lotus Notes

Enter any comments below:

No problem

8. Click OK. A message tells you that the request is sent to the next approver, Martha O’Reilly.


Expense1 Approval

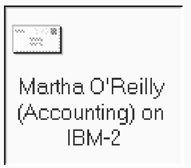
Notification has been sent to Martha O'Reilly.

9. Click OK to leave the message box.
10. Close your mailbox.

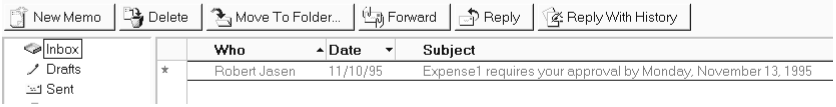
Working with the Request: The Accountant

You are now Martha O’Reilly, the accountant.

1. Open your mailbox.

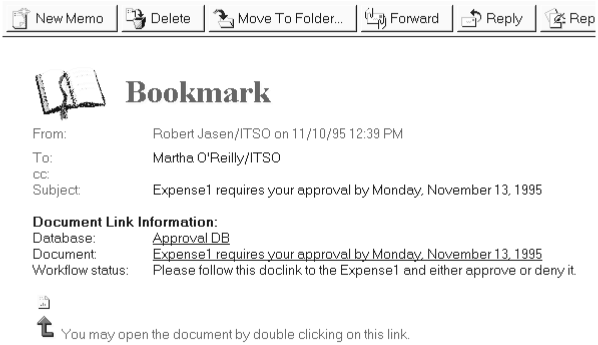


2. Open the request mail from the submitter Robert Jason.



	Who	Date	Subject
★	Robert Jasen	11/10/95	Expense1 requires your approval by Monday, November 13, 1995

3. To open the approval request, double-click the document link icon pointed to by a red arrow.



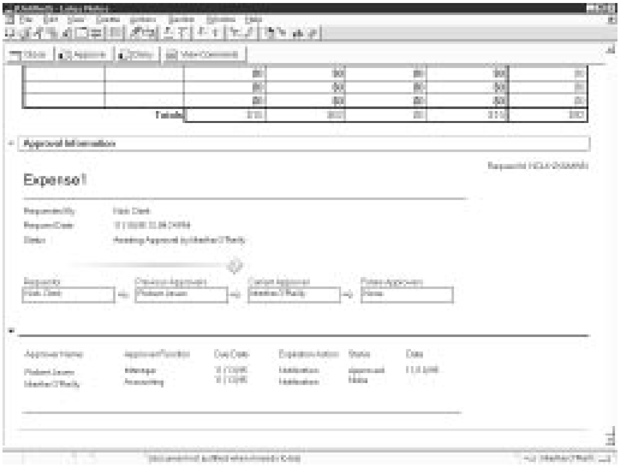
Bookmark

From: Robert Jasen/ITSO on 11/10/95 12:39 PM
To: Martha O'Reilly/ITSO
cc:
Subject: Expense1 requires your approval by Monday, November 13, 1995

Document Link Information:
Database: Approval DB
Document: Expense1 requires your approval by Monday, November 13, 1995
Workflow status: Please follow this doclink to the Expense1 and either approve or deny it.

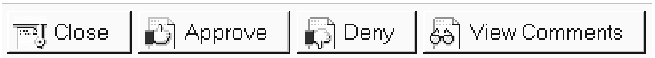
You may open the document by double clicking on this link.

4. To see the request-related information, move to the bottom of the request form and expand the Approval Information section if required.

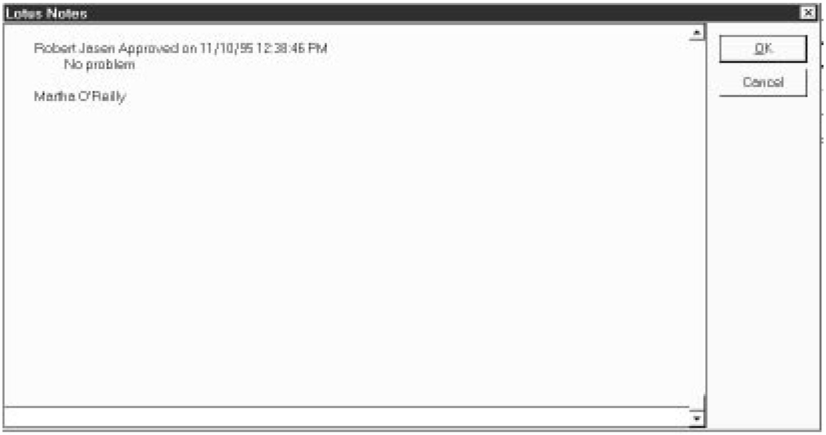


You can see that the previous approver has approved this request.

5. The action bar at the top of the window shows four buttons:



6. Click the View Comments button to see the comment added by the previous approver. It looks like this:

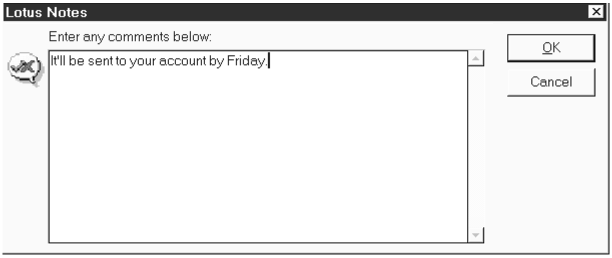


Robert Jasen Approved on 11/10/95 12:38:46 PM
No problem

Martha O'Reilly

7. Click OK to leave the comments box.

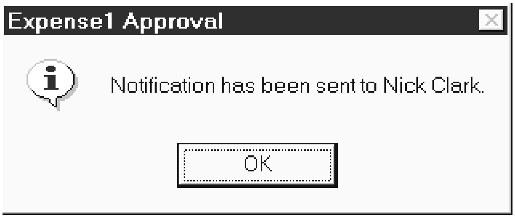
8. Click the Approve button and type your comments.



Enter any comments below:

It'll be sent to your account by Friday.

9. Click OK. A message tells you that the submitter of the request has been notified.



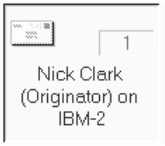
Notification has been sent to Nick Clark.

10. Click OK and close your mail.

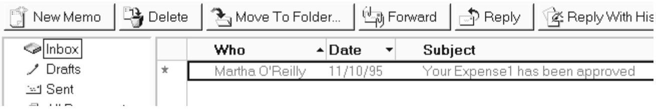
Checking the Status of the Request

You are again Nick Clark who submitted the request. Check the status of the request.

1. Open your mailbox.

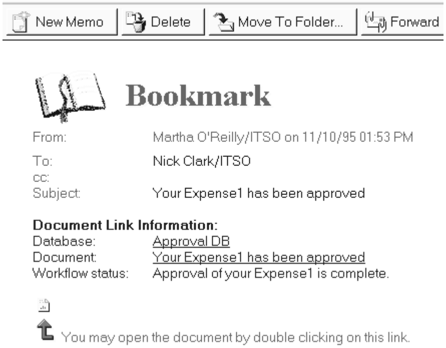


2. Open the mail sent from the accountant Martha O’Reilly.



	Who	Date	Subject
*	Martha O'Reilly	11/10/95	Your Expense1 has been approved

3. To open the approval request, double-click the document link icon pointed to by a red arrow.

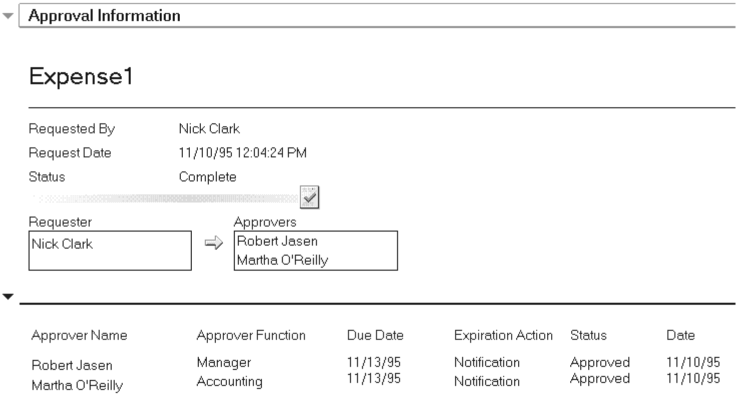


From: Martha O'Reilly/ITSO on 11/10/95 01:53 PM
To: Nick Clark/ITSO
cc:
Subject: Your Expense1 has been approved

Document Link Information:
Database: Approval DB
Document: Your Expense1 has been approved
Workflow status: Approval of your Expense1 is complete.

You may open the document by double clicking on this link.

4. Move to the bottom of the document and expand the two sections if required. The Status column shows that all approvers have approved your request.

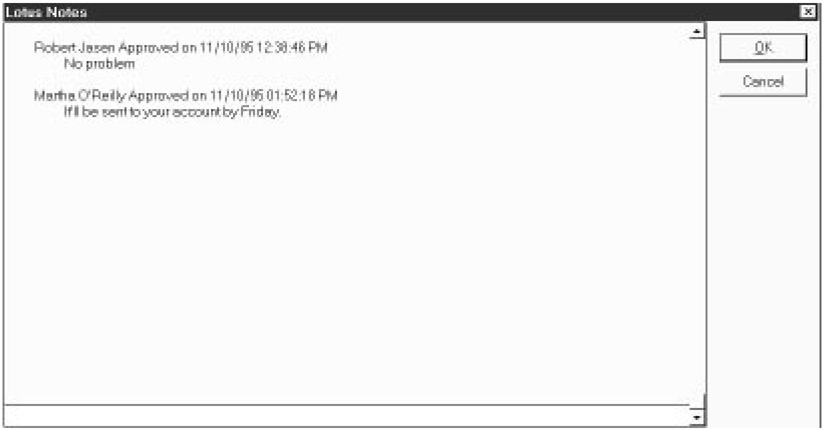


Requested By Nick Clark
Request Date 11/10/95 12:04:24 PM
Status Complete

Requester: Nick Clark
Approvers: Robert Jasen, Martha O'Reilly

Approver Name	Approver Function	Due Date	Expiration Action	Status	Date
Robert Jasen	Manager	11/13/95	Notification	Approved	11/10/95
Martha O'Reilly	Accounting	11/13/95	Notification	Approved	11/10/95

5. Click the View Comments button to see the comments added by the approvers:



Robert Jasen Approved on 11/10/95 12:38:46 PM
No problem

Martha O'Reilly Approved on 11/10/95 01:52:18 PM
It'll be sent to your account by Friday.

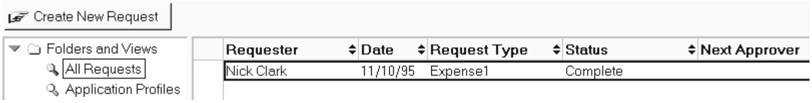
6. Click OK and close your mail.

You can also check the status of your request in the Approval DB database.

Note Users with reader access or above can check the status of their requests in the Approval database.

7. Open the Approval DB database.

8. Select the All Requests view in the navigator pane. You should see your request listed with a status of Complete.



Requester	Date	Request Type	Status	Next Approver
Nick Clark	11/10/95	Expense1	Complete	

9. Open the request and expand the Approval Information section to display more details on the request.

10. Click the View Comments button to see the comments of the approvers.

11. Click OK and close the database.

This completes our workflow example. The following sections give more detailed information on the design of the Approval Cycle database. Some useful hints and tips are also provided.

Approval Cycle Database: Design

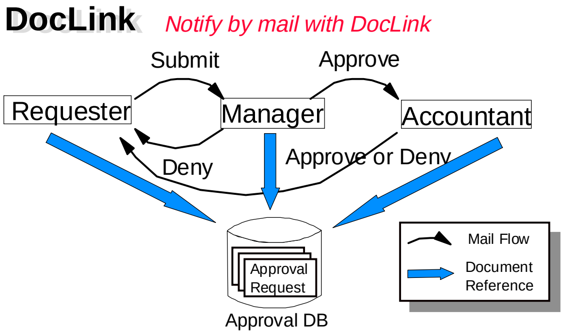
The following explains the design of the Approval Cycle database.

How Does a Form Flow?

There are two choices to process a request form: DocLink and Entire Document.

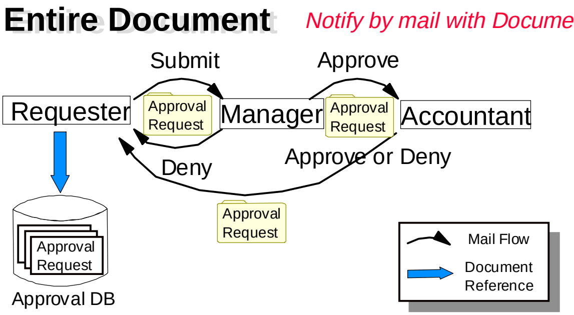
DocLink Form Flow

The following figure shows the DocLink type of form flow, which involves three users. All the notifications to process a flow are sent by mail containing a document link to an approval request document located in the Approval DB. The approvers can see the same document submitted by a requester for processing. Only the current approver can take an action (Approve or Deny) to deal with the request. The network path, server access authorization and appropriate database access controls are needed to share the same database for the workflow.



Entire Document Form Flow

The following figure shows the Entire Document type of form flow. An approval request form is sent to the mailbox of the next approver. The approval status in the Approval Database is automatically refreshed after an approver has processed a request.



How Is the Approval Cycle Database Organized?

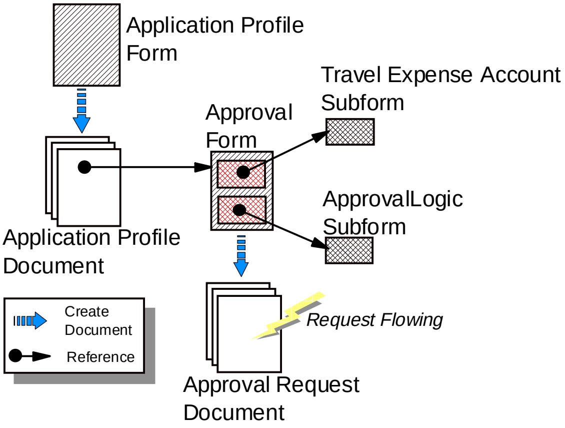
Relationship between Forms

The Approval Cycle template has two forms: The Application Profile Form and the ApprovalLogic Subform. It is recommended to create at least one document and one other form to use this template.

An Application Profile document must be created based on the Application Profile form to describe workflow properties and to specify an Approval form. The Approval form must include the ApprovalLogic subform and your workflow contents, for example, a Travel Expense Account.

An approval request document is created from the Approval form every time you start a workflow request. Approval status information is shown in the approval request document.

The following figure shows the relation between the forms:



Procedure-Calling Sequence and Event Handling

Approval Request Form

An Approval Request form consists of two subforms: The Approval Logic subform and the Travel Expense Account subform, which we added to the Approval Cycle template. The TEA subform contains some Notes macros, but no LotusScript programs. The ApprovalLogic subform contains a large number of LotusScript programs and many Notes macros, which control an approval request workflow.

The following picture shows part of the Approval Request document. It shows two dotted boxes. The first box is part of the Travel Expense Account subform, the second box is part of the ApprovalLogic subfom with a collapsed section.

The Travel Expense Account subform has some Notes features such as a layout region, option button, table, and others.

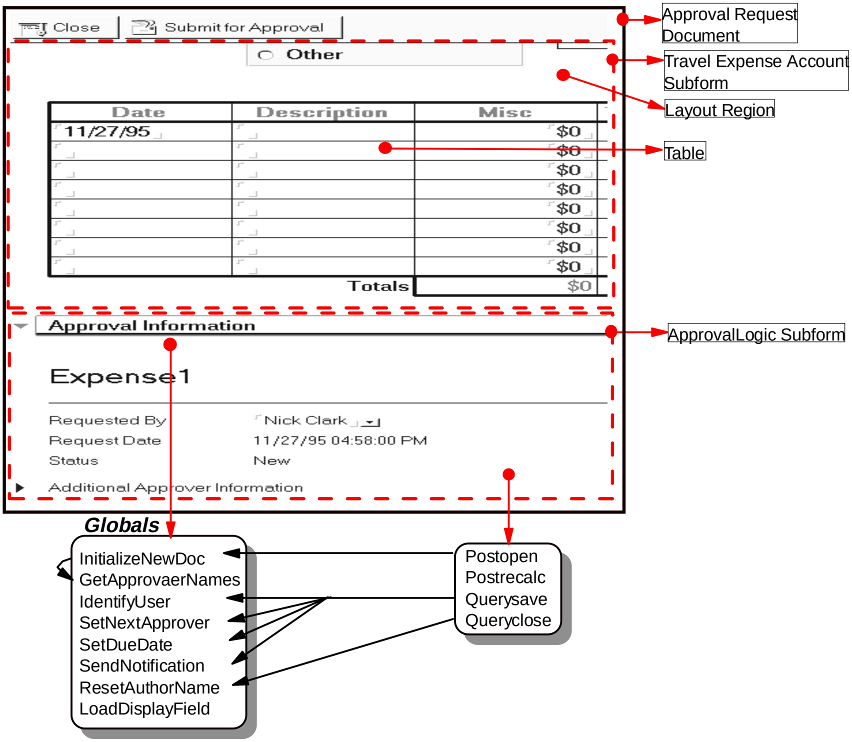
The ApprovalLogic subform has some event handlers and some global subroutines which are usually called in event handlers.

Some routines are shown in the shadowed box at the bottom of the picture. Some global routines are called by the Querysave event handler, which is performed when you click the Submit for Approval button shown at the top of the picture.

The Approval Logic subform does not have any user-defined classes and data types. All the variables and procedures are public, since Option Public is declared. Many variables are declared in global sections, and many implicit variables are used in scripts.

Note When you add procedures or modify scripts, be careful about global variables and implicit variables used in the ApprovalLogic subform to avoid conflicts of variables.

Tip If you write LotusScript programs, you should use Option Declare, user-defined classes and user-defined data types to make the programs clear and safe.



Date	Description	Misc
11/27/95		$0
		$0
		$0
		$0
		$0
		$0
		$0
		$0
	Totals	$0

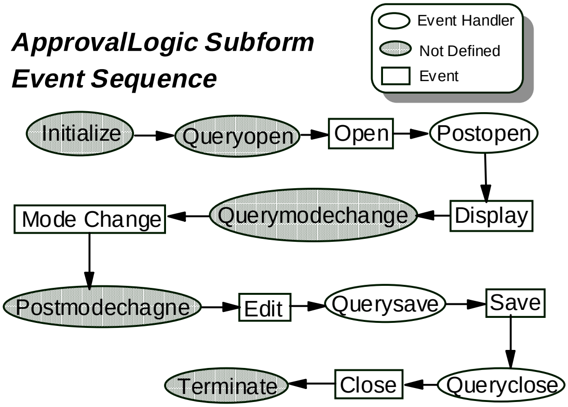
Expense1

Requested By Nick Clark
Request Date 11/27/95 04:58:00 PM
Status New

Additional Approver Information

When you created the Approval Request document, some events which are defined in the ApprovalLogic subform are shown in the following picture. This is because no event handlers are defined in either the Approval Request form or the Travel Expense Account subform.

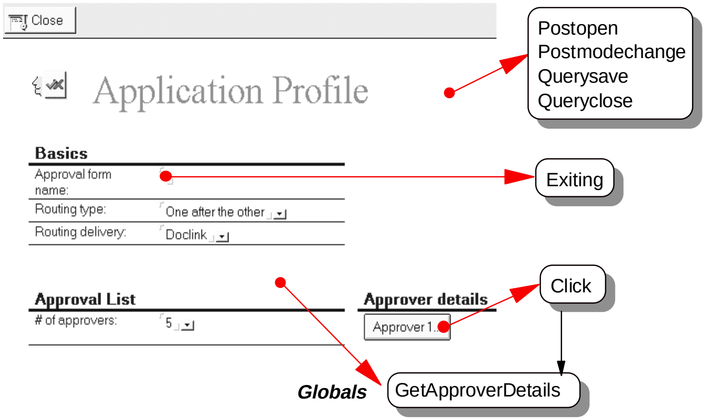
Eclipses displayed in gray are not performed because there are no scripts defined for them.



Note You can use the Notes debugger to see the sequence of real-time events. For more information on the debugger, see the chapter on Programming in Notes.

Application Profile Form

The Application Profile form has some event handlers, which are included in a form, a form global, fields and buttons. The Exiting event is performed when you try to move the cursor from the Approval form name field to another field, as shown in the following figure:



Close

Application Profile

Basics

Approval form name:

Routing type: One after the other

Routing delivery: Doclink

Approval List

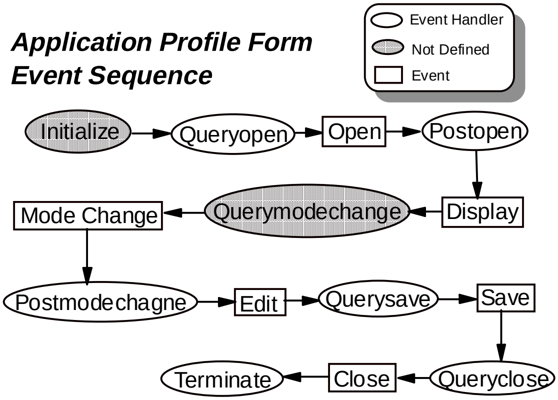
# of approvers: 5

Approver details

Approver 1...

You cannot move the cursor without typing a name, because the Exiting event handler checks the field contents. The Approver button has a Click event handler which is called by a mouse click trigger. Its handler calls the global routine GetApproverDetails, which displays a dialog box created by a layout region in the (Approverinfo) form.

The following picture shows an event sequence which occurs when you create the Application Profile document. Eclipses shown in gray are not performed, because there is no program defined for them.



Application Profile Form Event Sequence

You can use the Notes debugger to see the sequence of real-time events. For more information on the debugger, see the chapter on Programming in Notes.

Approval Cycle Database: Agent

The approval application requires an agent to deal with the due date expiration in this example. When the due date has passed and the approver has not taken any action, the agent processes the approval request depending on the criteria specified in the application profile for the due date.

Some Ideas on Adding Features to the Approval Database

Here are some ideas on possible additions to the approval database:

- Approval delegation may be useful, in case some approvers are out of the office and cannot process approval requests. You could add a feature that lets approvers delegate approval responsibility to other users.
- Another idea is to make it possible for approval requesters to cancel their requests before or during the workflow. This is not implemented in the template right now.

Chapter 11
Working With Lotus Components

Overview

By the end of this chapter you will know:

- What the Lotus Components are.
- How to create and access them using LotusScript.
- How to modify their properties using LotusScript.
- How to exchange data between a Component and Notes using Notes/FX
- How to link the Spreadsheet and Charting Component
- How to use NotesFlow publishing
- How to use LotusScript with Component events.
- How to create and package your own Components.

What Are Lotus Components?

Lotus Components are small, fast, reusable software applets that significantly extend the application possibilities for Lotus Notes Release 4. They may be used on-the-fly by Notes end users or embedded by Notes developers to create custom solutions.

On a technical level, Lotus Components are embeddable software modules contained within the Notes container. They're based on ActiveX (OCX) embedding technology, which takes full advantage of the 32-bit architecture of Notes 4.1 or higher. They share a consistent look and feel with the Lotus User Interface (UI) including context-sensitive menu integration and support for SmartIcons, InfoBox, live Status Bar, and right mouse-click menus. Lotus Components also take advantage of specific Notes features — such as LotusScript, Notes/FX 2.0, and security. In short, they are designed to extend the functionality of Lotus Notes for both Notes users and Notes application developers.

The initial delivery of Lotus Components is a collection of focused business applets called the Lotus Components Starter Pack. The Starter Pack includes the following:

- Lotus Spreadsheet Component
- Lotus Chart Component
- Lotus File Viewer Component
- Lotus Project Scheduler Component
- Lotus Draw / Diagram Component
- Lotus Comment Component
- Lotus Component Template Builder

The Template Builder application allows Notes application developers and experienced Notes users to apply business logic and formatting to one of the above components and save it as a new component.

Typically, each component is less than 1 megabyte in size and has an activation time of a few seconds from a single mouse click.

Who Are They For?

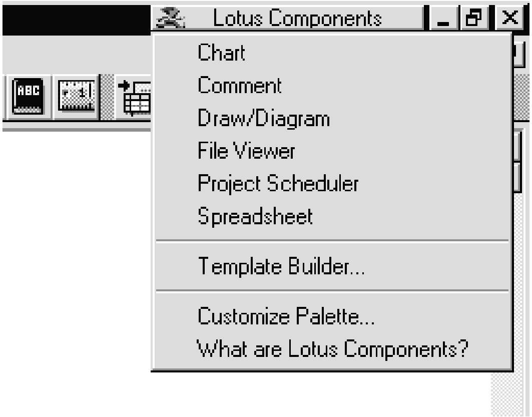
Lotus Components represent an exciting new technology that provides benefits to Notes application developers and end users.

Notes Application Developers

Lotus Components extend the functionality within a Notes application development environment, making it easier and faster to develop richer Notes applications.

A Lotus Component can be embedded into a form in a number of ways: first, by using the Lotus Components Palette which is added to the Notes title bar when the Lotus Components are installed; second, by embedding a component into a form at design time; and last, by embedding a component through LotusScript.

After installing the Starter Pack the Lotus Component Palette will look like this:



To use a Lotus Component, simply move the cursor to the rich-text field on the document where you would like to insert the Component, click on the Lotus Component Palette and select the Component you would like to embed into your document.

Lotus Components are Notes/FX-enabled so developers can bi-directionally exchange information between a Notes document and a Lotus Component. Lotus Components also have a wide range of predefined properties and methods, allowing the Notes application developer using LotusScript to precisely define how a Lotus Component looks and responds within a Notes application.

End Users

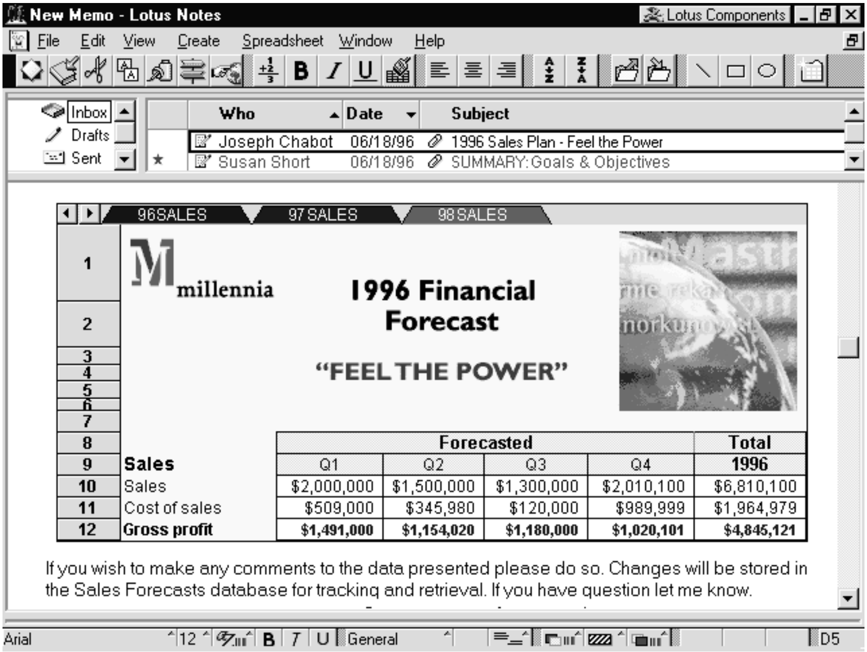
Notes users frequently require access to productivity tools while working in a Notes application — they collaborate on forecasts, review documents and display presentations — all usually requiring the time-consuming process of attaching and launching large documents. Lotus Components make it easier and faster for users to complete everyday business tasks while remaining in the familiar Notes environment.

Notes end users can easily use a Lotus Component on an as-needed basis by visually selecting the component they want, from within the Notes desktop, via the Lotus Component Palette. While working with the Lotus Component the user remains in the Notes environment, using the InfoBox, SmartIcons and the status bar. Users leverage their existing knowledge of working within Notes and require little training in the use of the component. When a user receives a Notes document with a Lotus Component, it quickly activates with a single click — no detaching or launching.

Lotus Spreadsheet Component

The Lotus Spreadsheet Component allows users to calculate and present numerical data within Lotus Notes Release 4.1. For Notes applications developers, it provides a tool to quickly embed powerful calculation functions into Notes applications.

This is how the Lotus Spreadsheet Component looks, after some data has been typed in and the cells formatted with color and different font styles.



	Who	Date	Subject
	Joseph Chabot	06/18/96	1996 Sales Plan - Feel the Power
★	Susan Short	06/18/96	SUMMARY: Goals & Objectives

8		Forecasted				Total
9	**Sales**	Q1	Q2	Q3	Q4	**1996**
10	Sales	$2,000,000	$1,500,000	$1,300,000	$2,010,100	$6,810,100
11	Cost of sales	$509,000	$345,980	$120,000	$989,999	$1,964,979
12	**Gross profit**	**$1,491,000**	**$1,154,020**	**$1,180,000**	**$1,020,101**	**$4,845,121**

If you wish to make any comments to the data presented please do so. Changes will be stored in the Sales Forecasts database for tracking and retrieval. If you have question let me know.

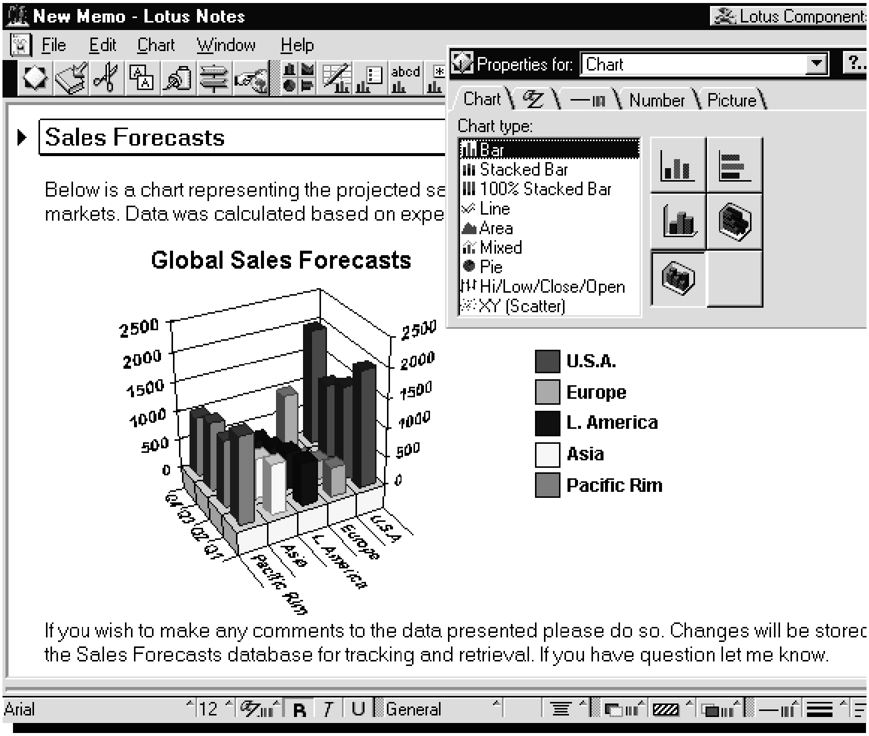
Tip Note how the menu bar, SmartIcons bar and status bar change to reflect the options available to the component you are currently using.

Highlights:

- User interface integrated with Lotus Notes Release 4: context-sensitive menus, live Status Bar, InfoBox, and SmartIcons
- 3D spreadsheet includes up to 256 worksheets x 256 columns x 16,000 rows
- Create, import, and export Lotus 1-2-3 and MS Excel worksheets
- Customizable AutoFill feature to build spreadsheets quickly
- Relative and absolute cell references
- Automatic formula syntax checking

Lotus Chart Component

The Lotus Chart Component allows users to graphically depict information from within a Notes document or a Notes application.



Sales Forecasts

Below is a chart representing the projected se
markets. Data was calculated based on expe

If you wish to make any comments to the data presented please do so. Changes will be storec
the Sales Forecasts database for tracking and retrieval. If you have question let me know.

Properties for: Chart

Chart type:

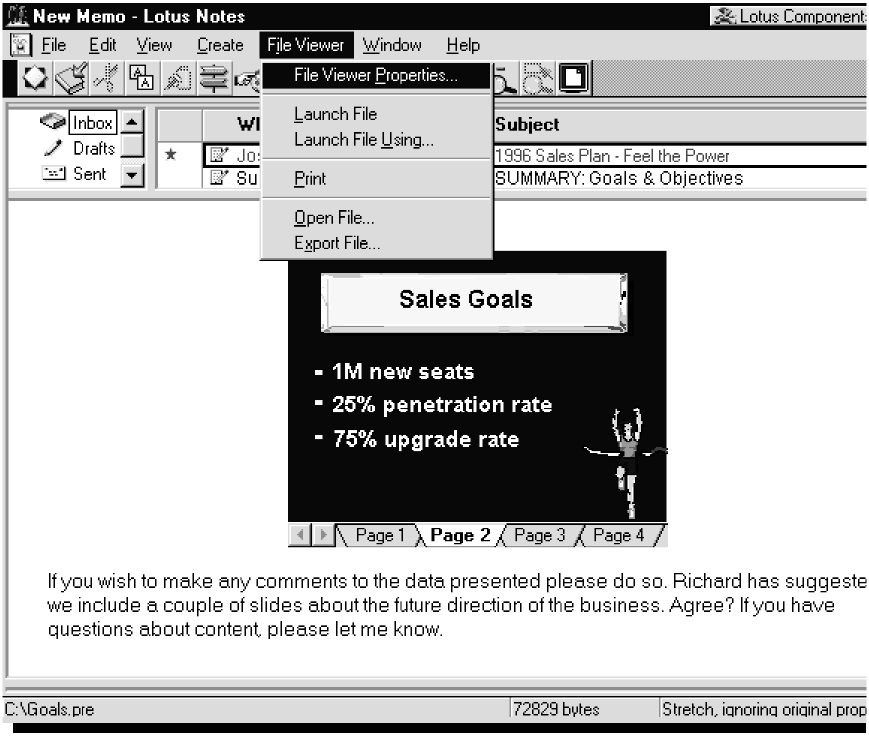
Highlights:

- User interface integrated with Lotus Notes Release 4: context-sensitive menus, live Status Bar, InfoBox, and SmartIcons
- Dynamic linking to the Lotus Spreadsheet Component
- A Photo-Realistic rendering engine allows charts to be drawn in true perspective
- Complete set of 2D and 3D chart types:
  - 2D and 3D horizontal bar
  - 2D and 3D vertical bar
  - 2D and 3D horizontal stacked
  - 2D and 3D vertical stacked
  - 2D and 3D 100% horizontal stacked
  - 2D and 3D 100% vertical stacked

  - 2D and 3D line
  - 2D and 3D area
  - 2D and 3D mixed bar, line, and area
  - 2D and 3D pie and multiple pie
  - High-Low-Close-Open
  - 2D xy (scatter diagram)
- Series smoothing feature in Line and Area charts

Lotus File Viewer Component

The Lotus File Viewer Component allows users to view files within Notes without launching the source application and opening the file. It streamlines communication since users do not need the source application installed on their PC to view files.



If you wish to make any comments to the data presented please do so. Richard has suggeste we include a couple of slides about the future direction of the business. Agree? If you have questions about content, please let me know.

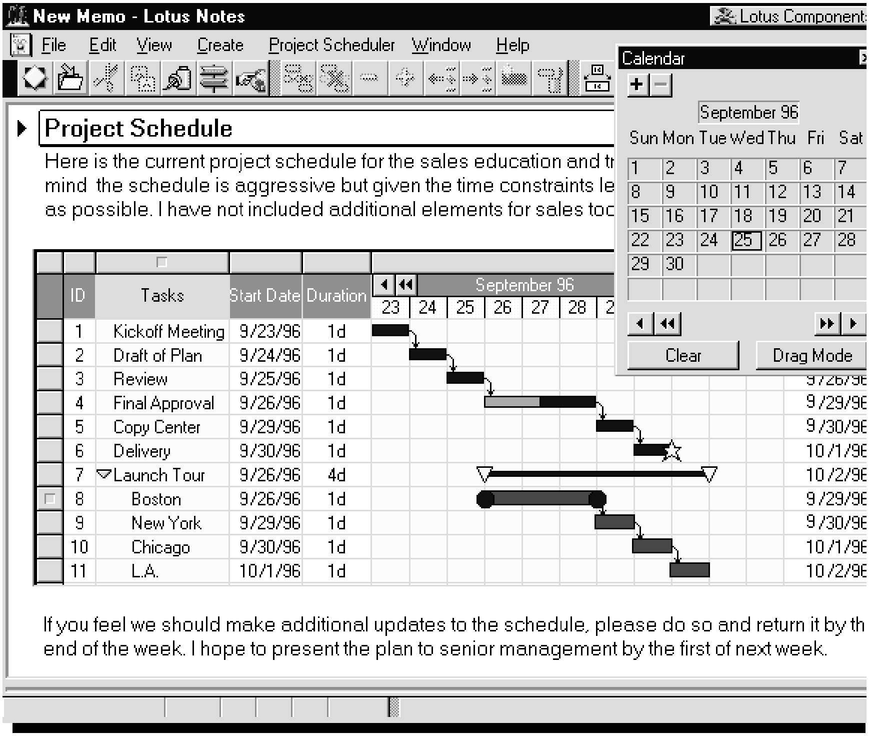
Highlights:

- User interface integrated with Lotus Notes Release 4: context-sensitive menus, live Status Bar, InfoBox, and SmartIcons
- Customizable sizing for Notes documents and applications
- Graphics scaling for optimal viewing
- Full zoom capabilities
- Supports dozens of the industry's most popular file types supported including:
  - Rich text format (.RTF)
  - Text (.TXT)
  - Bitmap (.BMP)
  - Computer Graphics Metafile (.CGM)
  - Windows Metafile (.WMF)
  - Lotus PIC
  - JPEG, TIFF (.TIF)
  - CompuServe GIF
  - Executable file (.EXE)
  - ZIP file (.ZIP)
  - Microsoft Access (.ACS)
  - Microsoft PowerPoint 4.x, 7.0 (PPT)
  - Microsoft Excel (.XL*)
  - Microsoft Word for Windows 6.0 (.DOC)
  - Lotus 1-2-3 for Windows (.WK3, .WK4)
  - Freelance Graphics (.PRE)
  - Ami Pro (.SAM)
  - Ami Draw (.SDW)
  - WordPerfect 6.x (.DOC)
  - WordPerfect Graphics 2.0
  - WordPerfect for Macintosh 2.0, 3.0
  - Word for Macintosh 4.0, 5.0
  - Macintosh PICT
  - Macintosh PICT2

File Viewer lets you view files in many common file formats. If a file format is not supported, File Viewer displays it as ASCII, if possible. If an ASCII display is not possible, a message appears indicating that the file type is not supported.

Lotus Project Scheduler Component

The Lotus Project Scheduler Component provides users with all the tools you need to track tasks and schedules and get a graphical snapshot of task relationships.



Project Schedule

Here is the current project schedule for the sales education and tr
mind the schedule is aggressive but given the time constraints le
as possible. I have not included additional elements for sales toc

ID	Tasks	Start Date	Duration	
1	Kickoff Meeting	9/23/96	1d	
2	Draft of Plan	9/24/96	1d	
3	Review	9/25/96	1d	9/26/9
4	Final Approval	9/26/96	1d	9/29/9
5	Copy Center	9/29/96	1d	9/30/9
6	Delivery	9/30/96	1d	10/1/9
7	Launch Tour	9/26/96	4d	10/2/9
8	Boston	9/26/96	1d	9/29/9
9	New York	9/29/96	1d	9/30/9
10	Chicago	9/30/96	1d	10/1/9
11	L.A.	10/1/96	1d	10/2/9

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

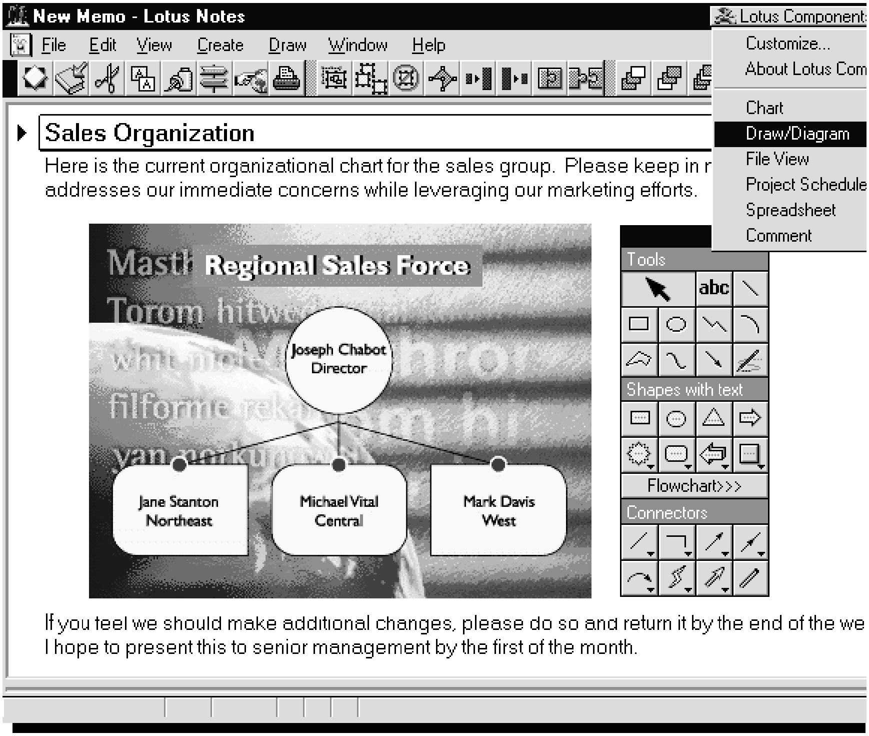
If you feel we should make additional updates to the schedule, please do so and return it by th
end of the week. I hope to present the plan to senior management by the first of next week.

Highlights:

- User interface integrated with Lotus Notes Release 4: context-sensitive menus, live Status Bar, InfoBox, and SmartIcons
- Spreadsheet-style interface for easy data entry and viewing
- Graphical Gantt Scale Column
- Fully adjustable columns for customized schedules
- Complete selection of time units: hours, days, weeks, months, quarters, and years
- Data entry via calendar feature — drag and drop
- Linking of related tasks
- Expandable and collapsible hierarchy of columns
- Insert and delete columns
- Customizable grid lines and styles

Lotus Draw/Diagram Component

The Lotus Draw / Diagram Component allows users to create professional drawings and diagrams such as timelines, organizational charts, and flow charts within Notes.



Sales Organization

Here is the current organizational chart for the sales group. Please keep in r
addresses our immediate concerns while leveraging our marketing efforts.

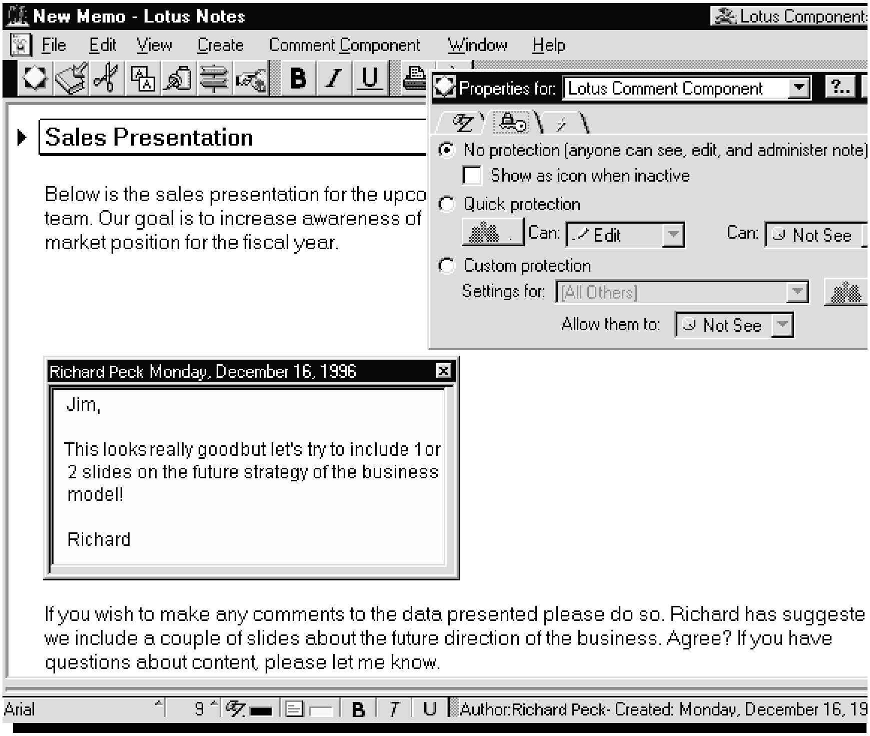
If you feel we should make additional changes, please do so and return it by the end of the we
I hope to present this to senior management by the first of the month.

Highlights:

- User interface integrated with Lotus Notes Release 4: context-sensitive menus, live Status Bar, InfoBox, and SmartIcons
- Align, flip and rotate commands
- Dynamic connectors linking drawn objects — as objects are moved, connecting lines move with them
- Selection of text shapes
- Libraries of pre-designed diagrams and clip art
- Edit points mode
- Powerful text manipulation
- Grid snapping

Lotus Comment Component

The Lotus Comment Component allows users to add a comment to a Notes document, business proposal, or presentation. To aid collaboration, the Lotus Comment Component identifies who created the comment and the time/date it was created. In addition, by using Lotus Notes security, it is possible to restrict the access of your comments to a certain group of people that you specify.



Sales Presentation

Below is the sales presentation for the upco
team. Our goal is to increase awareness of
market position for the fiscal year.

Properties for: Lotus Comment Component

● No protection (anyone can see, edit, and administer note)

□ Show as icon when inactive

○ Quick protection

Can: Edit Can: Not See

○ Custom protection

Settings for: [All Others]

Allow them to: Not See

Richard Peck Monday, December 16, 1996

Jim,

This looks really good but let's try to include 1 or 2 slides on the future strategy of the business model!

Richard

If you wish to make any comments to the data presented please do so. Richard has suggeste
we include a couple of slides about the future direction of the business. Agree? If you have
questions about content, please let me know.

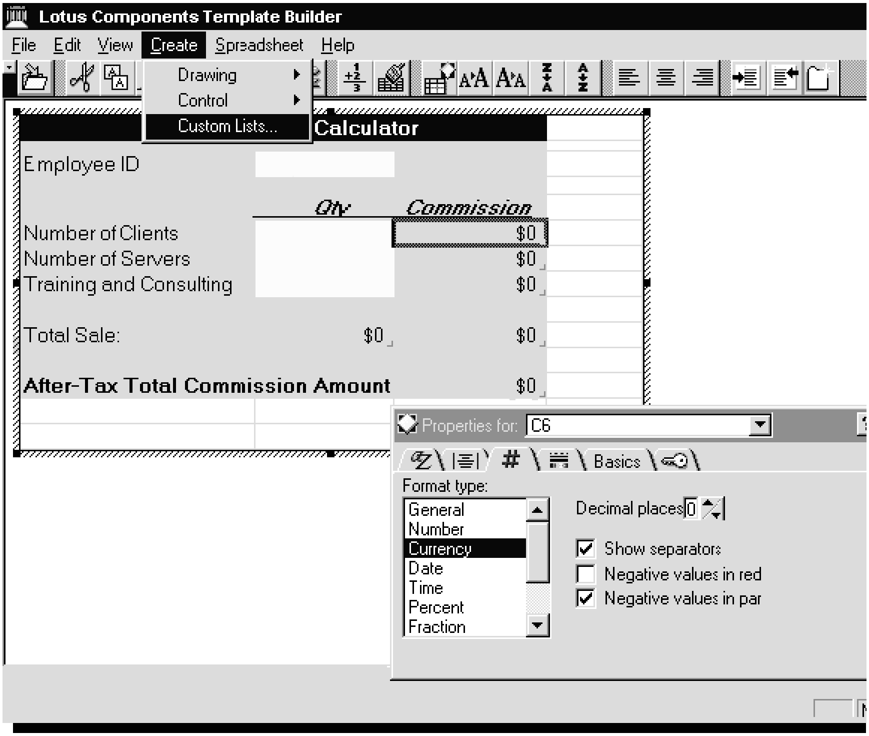
Arial 9 B I U Author:Richard Peck- Created: Monday, December 16, 19

Highlights:

- User interface integrated with Lotus Notes Release 4: context-sensitive menus, live Status Bar, InfoBox, and SmartIcons
- Single-click activation
- Security capabilities to assign workgroup editing rights including: no access, depositor, reader, editor and manager access.
- Automatic sizing
- Expandable and collapsible view
- Title strip display showing author, date and time

Lotus Components Template Builder

The Lotus Components Template Builder allows users and developers to customize Lotus Components for their specific business needs. For example, with the Lotus Template Builder, users and developers can convert the Lotus Spreadsheet Component into a Sales Commission Calculator for use throughout an organization. Customized Lotus Components can be used by users or embedded into Notes applications.


Lotus Components Template Builder
File Edit View Create Spreadsheet Help
Drawing
Control
Custom Lists...
Properties for:
Basics

Calculator

	Qty	Commission
Employee ID		
Number of Clients		$0
Number of Servers		$0
Training and Consulting		$0
Total Sale:	$0	$0
After-Tax Total Commission Amount		$0

Properties for: C6

Format type: General, Number, Currency, Date, Time, Percent, Fraction

Decimal places 0

☑ Show separators

□ Negative values in red

☑ Negative values in par

Note By building on the standard Lotus Spreadsheet Component you can add your own formulas, titles, text and color and create your own custom Component templates. These custom templates can then be distributed to other users and added to the Lotus Components Palette.

Using Lotus Components With LotusScript

Each of the Lotus Components has a rich set of LotusScript class extensions that the Notes application developer can use to control the way in which each component is created, displayed and manipulated.

For an introduction into LotusScript, read the earlier chapter in this book on Application Development.

Note There is a redbook available entitled *LotusScript for Visual Basic Programmers* (Lotus Part Number:12498 or IBM Form Number 5624-4856).

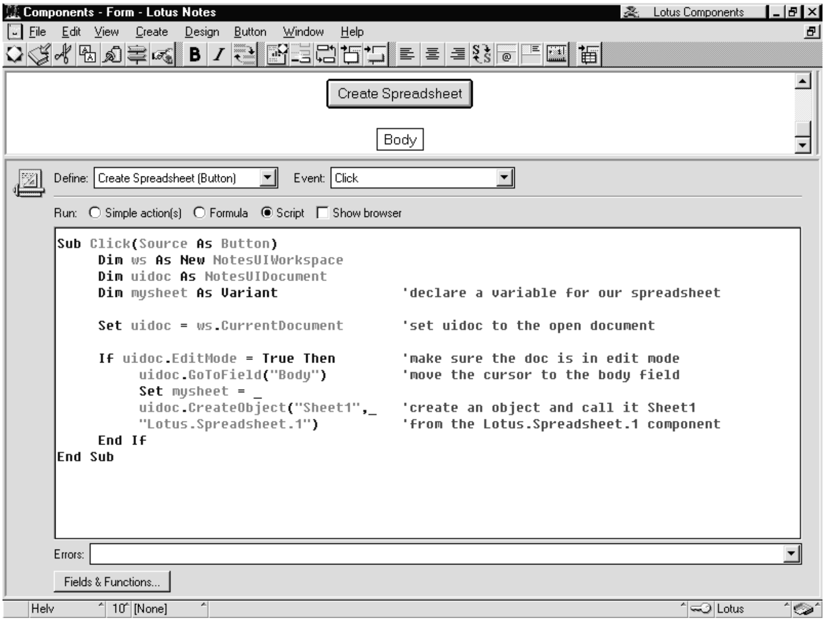
Adding Lotus Components to a Form Using LotusScript

In the following example we will add a spreadsheet component to a document by clicking on a button.

1. Firstly, create a new form by selecting Create-Design-Form from the menu.
2. Add a button to the form by selecting Create-Hotspot-Button from the menu.

   Give this button a name and a title.
3. Add a field to the form by selecting Create-Field. Give this field a name of Body and select Rich-Text from the Type drop-down listbox.

   This is how the form will look in the design pane when it is completed.



Define: Create Spreadsheet (Button) Event: Click

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

```
Sub Click(Source As Button)
     Dim ws As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Dim mysheet As Variant                'declare a variable for our spreadsheet

     Set uidoc = ws.CurrentDocument        'set uidoc to the open document

     If uidoc.EditMode = True Then         'make sure the doc is in edit mode
          uidoc.GoToField("Body")          'move the cursor to the body field
          Set mysheet = _
          uidoc.CreateObject("Sheet1",_    'create an object and call it Sheet1
          "Lotus.Spreadsheet.1")           'from the Lotus.Spreadsheet.1 component
     End If
End Sub
```

Errors:

4. Now let's look at the code. First, click on the button and make sure that you have the Click event displayed in the design window.

   Declare three variables in memory, the first from the NotesUIWorkspace class so that we can get access to the current document displayed on the screen and assign it to the second variable, uidoc. The third variable is declared as a Variant type to hold the Lotus Component object.

   The next line down assigns the value of uidoc to the document open on the screen.

   Next, we do a very simple error check to ensure that the document is actually open in edit mode by calling the uidoc.EditMode routine. This error check is important as a Lotus Component cannot be added to a form if it is not in edit mode.

   If the document is in edit mode we move the cursor to the Body field otherwise we go to the end of the program. An alternative here would be to force the document into edit mode using the command:

   ```
   uidoc.EditMode = True
   ```

   Now comes the call to uidoc.CreateObject to actually create the Lotus Spreadsheet Component.

   ```
   Set mysheet = uidoc.CreateObject("Sheet1",_
    "Lotus.Spreadsheet.1")
   ```

   The syntax for CreateObject is

   ```
   Set handle =
   notesUIDocument.CreateObject([name$],type$[,filePath$]]])
   ```

   The first parameter of CreateObject is an optional name parameter that you can use to refer to the object at a later date by using the GetObject call. The second parameter is the name of the Lotus Component entry stored in the OLE registry. The third parameter is not used when embedding a Component.

The following table lists the available types for each of the Lotus Components in the Starter Pack.

Lotus Component Types

Lotus Component Name	*Type*
Lotus Chart Component	Lotus.Chart.1
Lotus Comment Component	Lotus.Comment.1
Lotus Draw / Diagram Component	Lotus.Draw.1
Lotus Project Component	Lotus.Project.1
Lotus FileViewer Component	Lotus.FileViewer.1

If two Lotus Components of the same type are added to a document, for example two Spreadsheet Components, both of the Lotus Components will have the same name. This can be problematic if you need to gain access to either one of the Lotus Components programmatically, because both the Lotus Components would have shared the same name. If you do not assign a name to the component, then it assumes the full Lotus Component name in column one of the above table, for example, a spreadsheet component becomes “Lotus Spreadsheet Component,” so to get a handle to the Component you would use the following call:

```
Set mysheet = uidoc.GetObject("Lotus SpreadSheet Component")
```

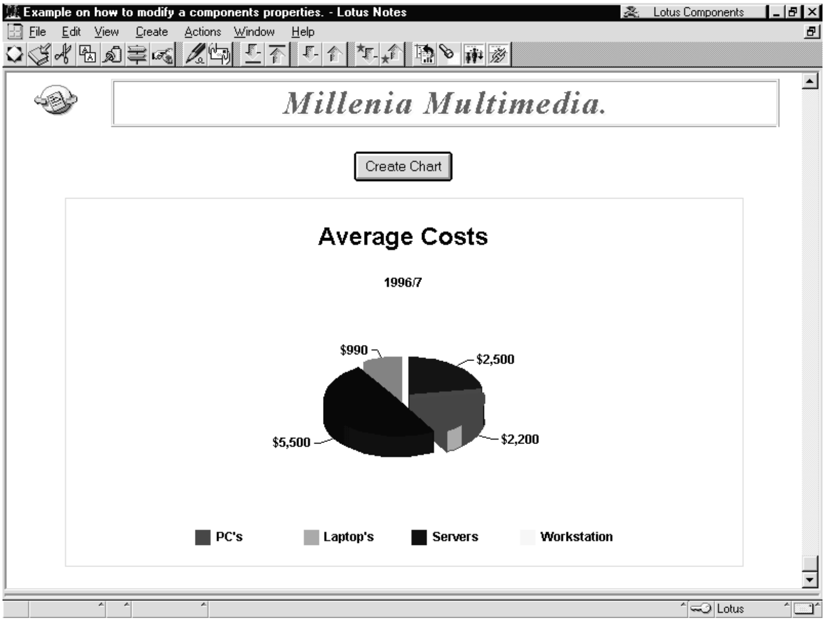
Setting and Modifying Properties

Each of the Lotus Components has a wide range of properties that can be used to control the way that data is displayed as well as numerous other features.

In the following example we will create a Chart Component from a button and set some of its properties by following these steps:

- Set the chart type to a 3d pie chart
- Set a title for the chart and the series
- Set a maximum data range
- Fill the data table with values and titles
- Display the legend at the bottom of the screen
- Display each segments value on the chart
- Explode each segment in the chart

This is how the final chart will look after the button has been clicked:



Now, create a new form and create a button and a rich-text field as before.

The following is the source code used to create the above chart from the click event of the button. Pay special attention to the second line in the program that tells you where to put the charting constants file, *CHCONSTS.TXT*, in the program.

The example has been commented to describe what is about to happen in the next line of code.

```
Sub Click(Source As Button)
    '** You must include the %include "CHCONSTS.TXT" in the
    '** forms Globals - Options section.
     Dim ws As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Dim mychart As Variant
     Dim MyOffset As Single

    '** set uidoc to the open document
     Set uidoc = ws.CurrentDocument
    '** make sure the doc is in edit mode
     If uidoc.EditMode = True Then
         '** move the cursor to the body field
          uidoc.GoToField("Body")
         '** create an object and call it Chart1 from the
         '** Lotus.Chart.1 component
```

```
 Set mychart = uidoc.CreateObject("Chart1", _
   "Lotus.Chart.1")

'** turn off repaint for faster updates
 mychart.Repaint = False

'** Set the new chart type to a 3d Pie chart. The
'** constant CHChartType3dPie is in the file
'** CHCONSTS.TXT which must be included in the
'** Global options for the form.
 mychart.ChartType = CHChartType3dPie

'** Now we will set various properties of the chart
 With mychart
      '** Set the chart title
      .Title.Text = "Average Costs"
      '** make the legend visible...
      .Legend.Location.Visible = True
      ’** .. and put it at the bottom
      .Legend.Location.LocationType = _
      CHLocationTypeBottom
      '** set the maximum no of rows to 4
      .rowcount = 4
      '** set the maximum no of columns to 1
      .Columncount = 1
      '** make sure we are using column 1
      .Column = 1
      '** set the column label title
      .ColumnLabel = "1996/7"

      '** set the current row to 1
      .Row = 1
      '** set the value to 2500
      .Data = 2500
      '** set the label to PC's
      .RowLabel = "PC's"

      '** set up three more data points
      .Row = 2
      .Data = 2200
      .RowLabel = "Laptop's"

      .Row = 3
      .Data = 5500
      .RowLabel = "Servers"

      .Row = 4
      .Data = 990
```

```
               .RowLabel = "Workstation"
          End With

          '** Now we want to display the values for each
          '** segment on the chart, add a bent line between
          '** the value and the segment, reduce the font
          '** size, format the output value and explode the
          '** chart
          '** set a 0.1 cm/inch offset
          MyOffset = 0.1
          '** iterate through each segment
          For i = 1 To 4
          With
mychart.Plot.SeriesCollection.Item(i).DataPoints
                    .Item(1).DataPointLabel.LocationType = _
                       CHLabelLocationTypeOutside
                    .Item(1).DataPointLabel.LineStyle = _
                       ChLabelLineStyleBent
                    .Item(1).DataPointLabel.ValueFormat = _
                       "$#,##"
                    .Item(1).DataPointLabel.VTFont.Size = 9
                    .item(1).Offset = MyOffSet
               End With
          Next

          '** turn repaint back on to update all changes
          mychart.Repaint = True
     End If
End Sub
```

Using Notes/FX With Components

The Lotus Notes Field Exchange technology can be used to exchange data bi-directionally between a Lotus Component and Notes 4.X fields, making data in Lotus Components available in ways that with LotusScript alone is more difficult to obtain. Notes/FX is not a replacement for LotusScript, rather it can be used to enhance the LotusScript environment.

One of the main reasons to transfer data between a Lotus Component and Notes fields is to make data in the Lotus Component available for views. Since views cannot access the data in a Lotus Component, the data must be transferred to a Notes field which makes the data available for a view.

Conversely, you can use LotusScript to write a value to a field that exchanges information with a Lotus Component. The value written to the field will be passed to the Lotus Component when the Component is activated.

Note It is important to realize the value is not passed to the Lotus Component until it is refreshed. Refreshing the Lotus Component can be achieved either by making it active by a user single-clicking on it or via the DoNotesFX LotusScript method.

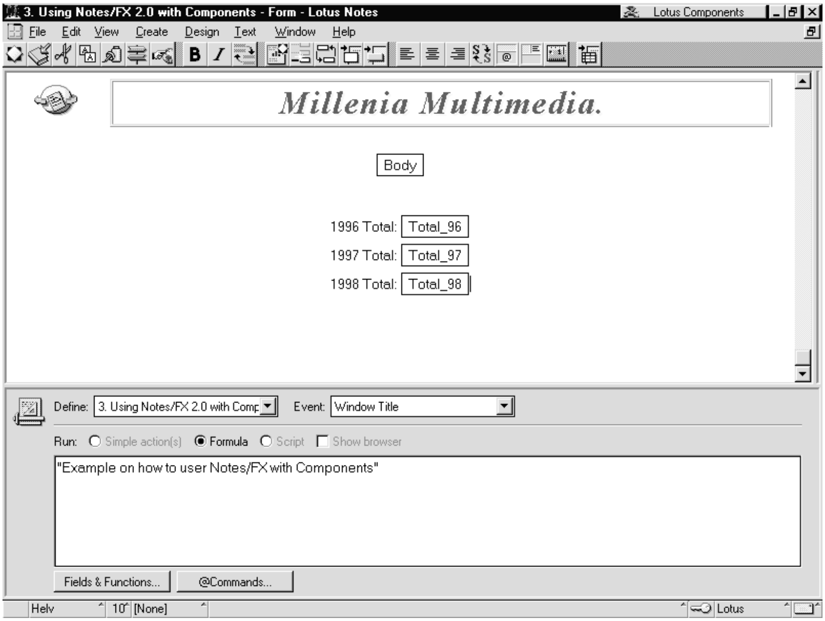
DoNotesFX is a Lotus Components LotusScript method that forces field exchange to take place.

Notes/FX Example

Notes/FX is now significantly easier to use than in previous versions. The following is an example of how to extract three values from a spreadsheet component and place them into Notes fields.

1. First, create a new form in your database and create four fields.
2. The field where the Component will be embedded needs to be a rich text field.
3. Create the other three fields to be editable currency fields.

   Following is a picture of how the form will look in the design pane.



3. Using Notes/FX 2.0 with Components - Form - Lotus Notes

Millenia Multimedia.

Body

1996 Total: Total_96

1997 Total: Total_97

1998 Total: Total_98

Define: 3. Using Notes/FX 2.0 with Comp

Event: Window Title

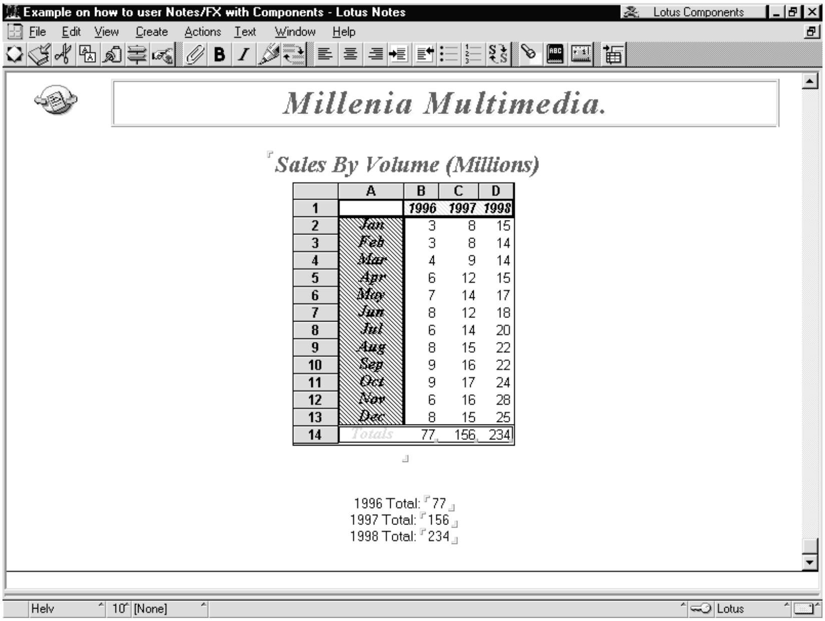
Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

"Example on how to user Notes/FX with Components"

4. When creating the names for the three currency fields, keep in mind that this is the name that the spreadsheet component will use to exchange information with. In our example we have called them Total_96, Total_97 and Total_98.

5. Next, save the form and create a new document from it using the Compose menu.

6. Embed a Lotus Spreadsheet component into your rich-text field by selecting Spreadsheet from the Lotus Components palette and format it to your liking. In our spreadsheet we have three columns representing the years 1996, 1997 and 1998 and rows for each month of the year. At the bottom of each column is a totals formula that adds together all the values for the year and it is these totals we are going to use Notes/FX with.


Example on how to user Notes/FX with Components - Lotus Notes
Lotus Components
File Edit View Create Actions Text Window Help
Helv
10
[None]
Lotus

Millenia Multimedia.

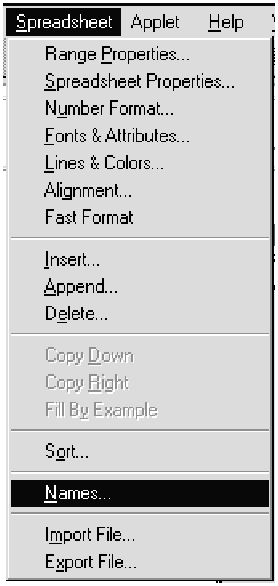
Sales By Volume (Millions)

	A	B	C	D
1		1996	1997	1998
2	Jan	3	8	15
3	Feb	3	8	14
4	Mar	4	9	14
5	Apr	6	12	15
6	May	7	14	17
7	Jun	8	12	18
8	Jul	6	14	20
9	Aug	8	15	22
10	Sep	9	16	22
11	Oct	9	17	24
12	Nov	6	16	28
13	Dec	8	15	25
14	Totals	77	156	234

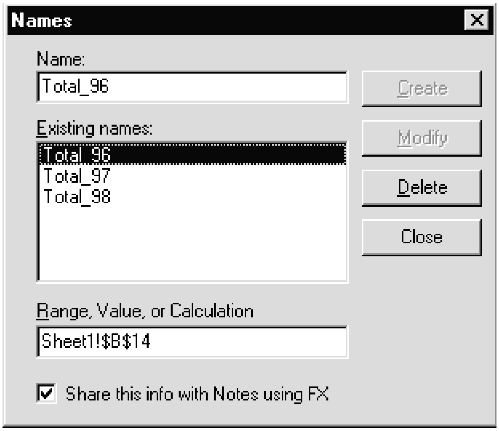
1996 Total: 77
1997 Total: 156
1998 Total: 234

7. To create the link between Notes and the Component, you need to tell the component which cell must exchange information with which Notes field.

Place your cursor in the spreadsheet cell you wish to exchange the information with and select Spreadsheet-Names from the menu bar.



8. In the Names dialog that is displayed, type into the Name field the name of the Notes field you created earlier. The name you type in here must be exactly the same as the Notes field you created on the form.


Names

Name: Total_96

Existing names:
Total_96
Total_97
Total_98

Range, Value, or Calculation: Sheet1!B14

☑ Share this info with Notes using FX

At the bottom of the dialog box you will find a reference to the cell that this name points to. In this case the name Total_96 refers to the cell B:14.

9. So far what you have done is name a spreadsheet range as you could with a Range-Name-Create command in Lotus 1-2-3. To enable the exchange of information to take place, check the box marked 'Share this info with Notes using FX.'

10. Repeat the above steps to create a range name for each of the other two currency fields, Total_97 and Total_98.

11. To test the field exchange, type in a new value into the spreadsheet component. When the total at the bottom of the column is recalculated with the updated figure, notice that the value in the Notes field also changes.

Note You cannot edit the value in a Notes field and use Notes/FX to update the value in the spreadsheet component if the cell you are trying to update contains a formula.

Linking the Spreadsheet and Charting Components

As with Lotus 1-2-3 you may have a need to display your spreadsheet information graphically using a chart.

With Lotus Components you can do this by setting four Components properties using LotusScript, one for the Spreadsheet Component and three for the Charting Component. They are as follows:

1. *spreadsheet.****TableName*** **=** *string*

   This assigns a name to the table that you can access through LotusScript and other external sources.

2. *chart.****ssLinkBook*** *= string*

   This sets the chart up to use the named spreadsheet that you set in the above item.

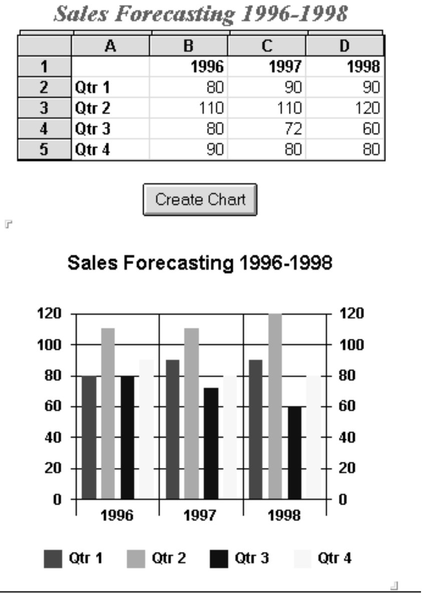
3. *chart.****ssLinkRange*** *= range*

   This allows you to define a range from the spreadsheet that will be charted. The range must be specified within quote marks, for example, mychart.ssLinkRange = "a1:b4".

This sets up the method by which the two components will refresh. Valid settings are:

No.	*Mode Name*	*Mode Description*
0	ChSsLinkModeOff	The connection to the spreadsheet is not active.
1	ChSsLinkModeOn	The spreadsheet is active. Lotus Chart Component makes no attempt to interpret the spreadsheet data. It uses the values set by the Column, Row, ColumnLabelCount, and RowLabelCount properties to determine the data grid dimensions and then fills those areas with data from the spreadsheet.
2	ChSsLinkModeAutoParse	The spreadsheet connection is active. Lotus Chart Component examines the spreadsheet data and tries to determine what is a label and what is data. It determines what it thinks the dimensions of the data grid should be and adjusts the values of the Column, Row, ColumnLabelCount, and RowLabelCount properties accordingly.

In the following example, a spreadsheet component has been embedded in a form at design time. When the user clicks the Create Chart button, a Charting Component is created and linked to the data within the spreadsheet. If a value in the spreadsheet is changed it is automatically refreshed in the chart.



Sales Forecasting 1996-1998

	A	B	C	D
1		1996	1997	1998
2	Qtr 1	80	90	90
3	Qtr 2	110	110	120
4	Qtr 3	80	72	60
5	Qtr 4	90	80	80

To create this example:

4. Create a new form.
5. Using the Lotus Component Palette, select the Spreadsheet Component.
6. Create a simple spreadsheet, like the one shown in the Notes/FX example, with columns for each quarter and three rows for the years.
7. Type some sample information into the table.
8. Create a button on the form with the code displayed next to the click event.
9. Create a rich text field and call it Body.

Below is the code used to create the chart shown. The program is commented to show you what is about to happen in the next line of code.

```
Sub Click(Source As Button)
     Dim ws As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     '** declare a variable for our chart
     Dim mychart As Variant
     '** declare a variable for our spreadsheet
     Dim mysheet As Variant

     '** set uidoc to the open document
     Set uidoc = ws.CurrentDocument

     '** make sure the doc is in edit mode
     If uidoc.EditMode = True Then
          '** move the cursor to the body field
          uidoc.GoToField("Body")
          '** create an object and call it Chart1
          '** from the Lotus.Chart.1 component
          Set mychart = uidoc.CreateObject("Chart1", _
          "Lotus.Chart.1")

          '** get a reference to the spreadsheet component
          Set mysheet = uidoc.GetObject( _
          "Lotus SpreadSheet Component")

          '** turn off the repaint for faster screen updates
          mychart.RePaint = False

          '** set the chart title and reduce the
          '** font slightly
          mychart.Title.VTFont.Size = 12
          mychart.TitleText = "Sales Forecasting 1996-1998"

          '** make the legend visible...
```

```
                    mychart.legend.location.visible = True
                    '** .. and put it at the bottom
                    mychart.legend.location.locationType = _
                    CHLocationTypeBottom

                    '** set the type of chart to a simple 2d bar chart
                    mychart.ChartType = ChChartType2dBar
                    '** set the maximum number of rows and columns
                    mychart.RowCount = 4
                    mychart.ColumnCount = 3

                    '** Name the spreadsheet "Forecast"
                    mysheet.TableName = "Forecast"
                    '** point the charts linkbook at the
                    '** "Forecast" spreadsheet
                    mychart.ssLinkBook = "Forecast"
                    '** set the range in the spreadsheet to chart
                    mychart.ssLinkRange = "a1:d5"
                    '** set the linkmode to "On"
                    mychart.ssLinkMode = 1

                    '** redraw the chart
                    mychart.RePaint = True
               End If
          End Sub
```

Using NotesFlow Publishing

Lotus Components use the NotesFlow Publishing capabilities of sharing commands together with specified Notes menu-merging conventions to facilitate seamless integration between Notes and Lotus Components. Developers can use NotesFlow Publishing to create applications that contain OLE objects and that isolate end users from jarring and potentially confusing user interface context switches. This means end users can easily perform Notes Actions when the embedded object has control of the User Inferface.

When an embedded ActiveX Control is activated by an end user, the object combines its own menu with that of Notes. The Notes Actions menu typically includes a set of developer-specified commands that are usually disabled when an object has the focus in a window. However, when NotesFlow Publishing is enabled for an Action you can make the action available to the active object as well as to Notes. Notes passes the action to the control using Notes/FX 2 technology, allowing the control to put these actions on its own actions menu.

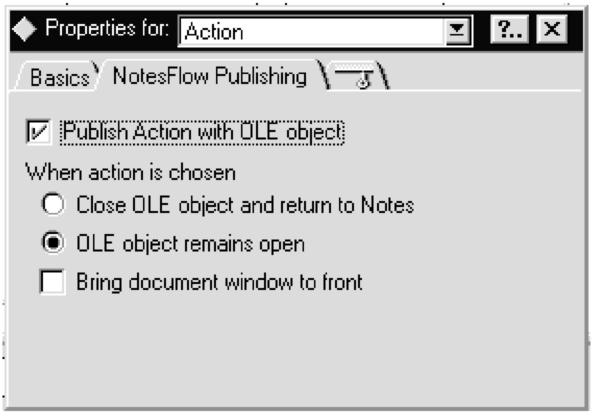
Action Publishing

Action Publishing uses Notes/FX 2 features to allow the application developer to publish Notes actions on the menu bar when a component is active.

An example of an action that you would like to have available at all times, to Notes and to Lotus Components, would be the *Forward* action. Making this action available to the Component means that the user does not have to click out of the Component they are working on in order to mail it to someone; they can simply click on Action-Forward in the menu bar.

To make an action available to a Lotus Component you need to do the following:

- Open the form where the action is that you would have appear on the menu bar when a Lotus Component is selected.
- In the InfoBox for the Action, select the NotesFlow Publishing Tab and check the box marked Publish Action with OLE object:



Action

☑ Publish Action with OLE object

When action is chosen

○ Close OLE object and return to Notes

● OLE object remains open

□ Bring document window to front

You also have control over what happens to the OLE object when you select this action.

- Close OLE Object and return to Notes will shift the focus away from the object, saving it if necessary and will return back to the Notes document.
- OLE Object remains open will keep the focus with the OLE object.
- Bring document window to front will allow you to keep the OLE object open and bring the current Notes document to the foreground. This enables you to get input from the user without closing the object.

Using LotusScript With Lotus Components Events

If you have used Lotus Components with Release 4 or 4.1 of Notes, you may have used the Action Linking feature. This enabled you to capture events from a Lotus Component, such as a mouse click, and pass it back to Notes triggering an action in response to it using the *ActionLinkSet/Get/Clear* statements.

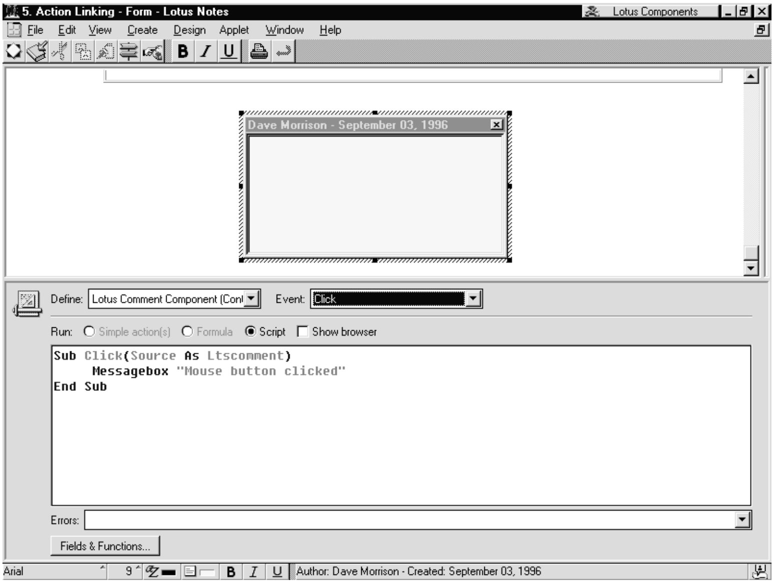
This has been greatly simplified in Release 4.5 of Lotus Notes. You no longer have to create an action or tell Notes the event that you want to trap, as all the events are now displayed to you in the programmer pane.

To make use of the new functionality you must embed a Lotus Component into the form at design time.

The following example shows you how this new feature of Notes works.

1. Create a new form in Lotus Notes by selecting Create-Design-Form from the menu bar.
2. In the design pane, select the Lotus Component that you wish to add to your form from the Lotus Components Palette.
3. Select the new Component with the mouse.
4. In the programmer's pane, notice that the define drop-down list box now contains the generic name for your new component. To the right of this, the Event drop-down list box lists all the events that are available to you.

   Note The list of events will differ depending on the component you chose to embed.



Define: Lotus Comment Component (Con
Event: Click

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

```
Sub Click(Source As Ltscomment)
     Messagebox "Mouse button clicked"
End Sub
```

Errors:

Author: Dave Morrison - Created: September 03, 1996

5. Add the LotusScript code for the event into the programmer's pane. For example, suppose we had embedded a Lotus Comment component into our form. On the click event we can pop up a message box that tells us that the mouse has been clicked:

```
Sub Click(Source as Ltscomment)
  MessageBox "Mouse Button Clicked"
End Sub
```

6. Save the form.
7. Test your form by creating a new document and clicking on the component. You should receive a messagebox with the text "Mouse Button Clicked."

Notice that in the above example, a handle to the component is automatically passed to Notes, in this case *Source as Ltscomment*. This can be different for each of the components you are using and the events you are coding for. For example, in the spreadsheet component if you trap the Click event you are passed a handle to the object, the row and the column.

Using the Lotus Components Template Builder

The Lotus Components Template Builder enables you to create and distribute your own Lotus Components, based on the existing components you have installed.

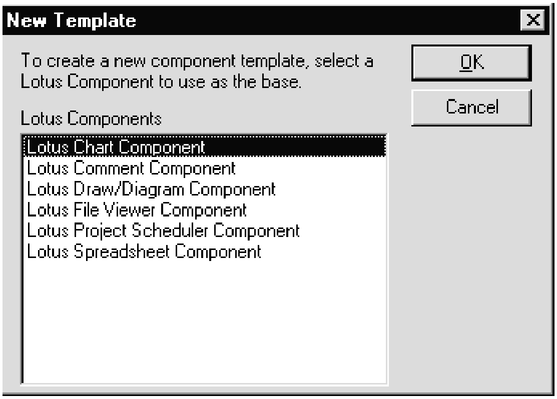
An example of these customized components may be a company expense form, where you take the spreadsheet component, add your own text, color and formulas and then package it as a new template that your employees can add to their documents from the Components Palette.

Creating Your Own Component

In the following example we will create a company expense form.

1. To access the Template Builder, select Template Builder from the Lotus Components Palette. The Lotus Component Template Builder program is displayed.

Select File-New Template from the menu. The New Template dialog box is displayed.



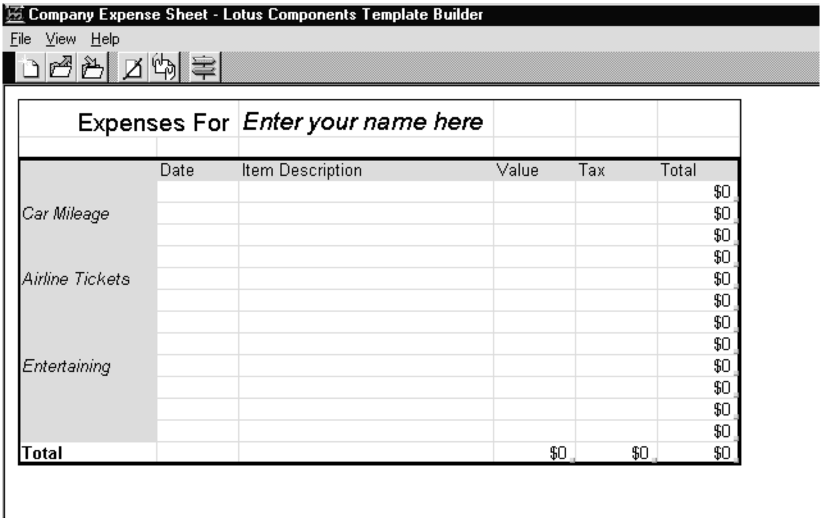
To create a new component template, select a Lotus Component to use as the base.

Lotus Components

- Lotus Chart Component
- Lotus Comment Component
- Lotus Draw/Diagram Component
- Lotus File Viewer Component
- Lotus Project Scheduler Component
- Lotus Spreadsheet Component

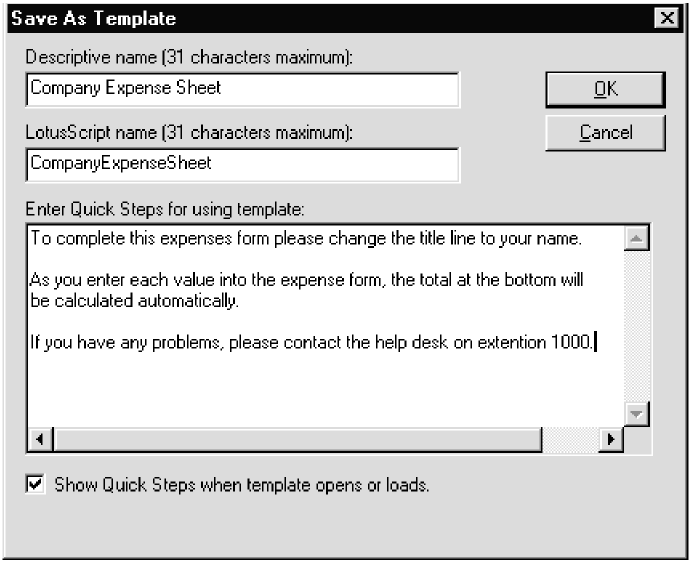
From the list of available components select the one that you will use as the base for your new component, in our case select the Lotus Spreadsheet Component and click OK.

2. In the Template Builder window a blank spreadsheet is displayed. You can now add text formatting and formulas to this spreadsheet to create your expense sheet. Below is a simple expense form we created:



Expenses For	*Enter your name here*				
	Date	Item Description	Value	Tax	Total
					$0
Car Mileage					$0
					$0
					$0
Airline Tickets					$0
					$0
					$0
					$0
Entertaining					$0
					$0
					$0
					$0
Total			$0	$0	$0

3. Once you are happy with the layout of your template, select File-Save from the menu. The Save-As-Template dialog box appears.



Descriptive name (31 characters maximum): Company Expense Sheet

LotusScript name (31 characters maximum): CompanyExpenseSheet

Enter Quick Steps for using template:

To complete this expenses form please change the title line to your name.

As you enter each value into the expense form, the total at the bottom will be calculated automatically.

If you have any problems, please contact the help desk on extention 1000.

☑ Show Quick Steps when template opens or loads.

Type in a descriptive name for the new component. This is the name that will appear in the Lotus Components Palette. Give the component a LotusScript name. This is used when accessing the component through LotusScript.

If you wish you can add a list of instructions that can be displayed to the user when they insert the component into a document.

4. Click OK to save the new component.

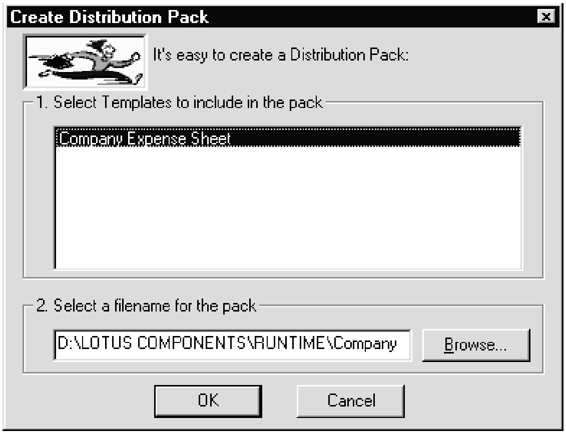
Creating a Distribution Pack

In order for your users to be able to use the new component, they must have the component installed on their workstation.

To help you with this task, the Lotus Component Template Builder comes with a tool called the Distribution Pack. This tool will create a self extracting installation program that will install the component on the users workstation, add the component into the Windows registry and display the component on the Notes Create-Object menu.

To create a distribution pack for our expense sheet, follow these steps.

1. Select File - Create Distribution Pack from the menu.



Create Distribution Pack

It's easy to create a Distribution Pack:

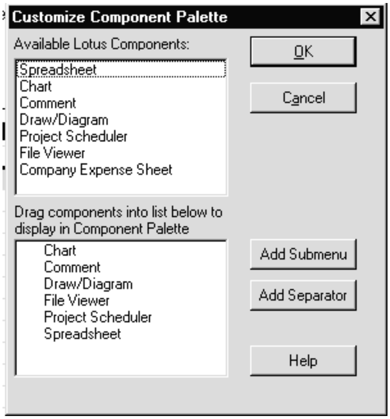
1. Select Templates to include in the pack

Company Expense Sheet

2. Select a filename for the pack

D:\LOTUS COMPONENTS\RUNTIME\Company

2. Select the templates that you wish to include in the distribution pack, (you can add more than one) and click OK.
3. The distribution pack is created and an informational dialog box appears.
4. You can now send this EXE file to those users that you wish to install the new component. After a user has installed the new component they must add it to their own Components Palette by selecting Customize Palette from the Lotus Components Palette.



Customize Component Palette

Available Lotus Components:

Spreadsheet
Chart
Comment
Draw/Diagram
Project Scheduler
File Viewer
Company Expense Sheet

Drag components into list below to display in Component Palette

Chart
Comment
Draw/Diagram
File Viewer
Project Scheduler
Spreadsheet

5. The Customize Component Palette contains two lists. The first is a list of all available components and the second is a list of components that appear on the Lotus Components Palette. To make our new template available we need to simply drag the entry from the top list into the second.

Chapter 12
Notes Applications and Security

This chapter will introduce Lotus Notes security from an application developer's viewpoint. The chapter is not intended to be an in-depth look at the full range of Notes security features, rather to provide the Notes application developer with the information they require to create secure applications.

In the chapter we will cover two areas:

- Access Control Lists (ACL) and Execution Control Lists (ECL)
- Security features within various design elements.

Access Control List

At the heart of every Notes database's security lies the Access Control List, commonly referred to as the ACL. This list contains information on the users and groups that can have access to the database, and the level these people are allowed to access at.

With Notes Release 4.5 there are seven main levels of access that a database administrator can assign to a person or group and eight sublevels below each of these.

Access Level	*Description*
Manager	Users with Manager access can modify ACL settings, encrypt a database for local security, modify replication settings, delete a database and perform tasks permitted by no other access level. Managers can also perform all tasks allowed by other access levels. Notes requires each database to have at least one Manager. It’s best to assign two people Manager access to a database in case one manager is absent.
Designer	Users with Designer access can modify all database design elements (fields, forms, views, public agents, the database icon, Using This Database document and About This Database document), can modify replication formulas, and can create a full text index. Designers can also perform all tasks allowed by lower access levels. Assign Designer access to the original designer of a database or to a user responsible for updating the design after a database is in use.
Editor	Users assigned Editor access can create documents and edit all documents, including those created by others. Assign Editor access, for example, to a user responsible for maintaining all data in a database.
Author	Users assigned Author access can create documents and edit documents they create. Assign Author access to allow users to contribute to a database but not edit documents created by others. When possible, use Author access rather than Editor access to reduce Replication or Save Conflicts.
Reader	Users assigned Reader access can read documents in a database but cannot create or edit documents. For example, assign Reader access to users who must be able to read the contents of a reference database such as a company policies database. Anyone with at least Reader access to a database can create personal agents in the database if the database manager selects the ACL option “Create personal agents.” However, users can only run agents that perform tasks allowed by their access levels. For example, someone with Reader access can create a private agent that deletes documents, but the agent won’t delete documents when the user runs it.
Depositor	Users assigned Depositor access can create documents but can’t see any documents in the database views, even the documents they create. For example, assign Depositor access to allow users to contribute to a mail-in database or to a database used as a ballot box.
No Access	Users assigned No Access cannot access the database. For example, assign No Access as the default access to prevent most users from accessing a confidential database.

As well as adding people or groups to a database ACL and assigning them an access level, you can also fine-tune their access by selecting or de-selecting certain sub-access options. Below is a table of these sub-levels.

Access Option	*Description*
Create Documents	Select this option to allow Authors to create documents. Managers, Designers, Editors, and Depositors are permanently assigned this access. You normally select this option for all users with Author access; you may want to deselect this option after a period of time to prevent Authors from adding any more documents but to allow them to read and edit ones they've already created. By default, the Create documents option is not selected for new Authors that you add to the access control list.
Delete Documents	Select this option to allow Managers, Designers, Editors, or Authors to delete documents. Authors can delete only documents that they created or if the document contains an Author Names field, they can delete documents if their name or a group or role that contains their name appears in the Author Names field.
Create Personal Agents	Select this option to allow Designers, Editors, Authors, or Readers to create personal agents. Managers are permanently assigned this access. Since personal agents on server databases take up server disk space and processing time, you may want to deselect this option to prevent some users from creating them. Note A Notes administrator can use the Agent Manager Restrictions section of a server document to prevent people from running personal agents on a server; people denied this server access can't create personal agents on the server, regardless of the ACL setting.
Create Personal Folders/Views	Select this option to allow Editors, Authors, or Readers to create personal folders and views in a database on a server. Managers and Designers are permanently given this access. Personal folders and views created on a server are more secure and available on multiple servers. Also, administrative agents can operate only on folders and views stored on a server. If this option is not selected, users can still create personal folders and views but the views and folders are stored on their local workstations. Deselect this option to save disk space on a server.
Create Shared Folders/Views	Select this option to allow Editors to create shared folders and views. Managers and Designers are permanently assigned this access. Deny this access to save disk space on a server and to maintain tighter control over database design.

continued

Access Option	*Description*
Create LotusScript Agents	Select this option to allow Readers, Authors, Editors or Designers to create LotusScript agents. Managers are permanently assigned this access. Since LotusScript agents on server databases have the potential to take up significant server processing time, you may want to restrict which users can create them. Note: A Notes administrator can use the Agent Manager Restrictions section of a Server document in the Public Address Book to prevent people from running restricted and/or unrestricted LotusScript agents on a server. If you select “Create LotusScript Agents” for a name in the ACL, a Server document can nevertheless prevent people assigned this access from creating LotusScript agents.
Read Public Documents	This option is to support the Calendaring and Scheduling delegation function. It enables an application developer to assign reader access to a database without giving the user the full Reader Role. A Notes application designer can add a $PublicAccess field to a document. When this field is in the document, it means that the document can be read by any user with Public Reader access and can be modified by anyone who has Public Writer access. A Notes application designer can check the [x] Available to Public access users checkbox on the Forms and Views Properties InfoBoxes to allow the object to be visible and usable by public access users.
Write Public Documents	See Read Public Documents above.

Note Some of these options are disabled depending on the level of access you have given a person. For example, if you give someone Manager access you will not be able to remove their ability to create documents.

Roles

Roles are very much back in fashion with Notes Release 4.5, having taken a back seat in earlier releases. A role is a tool by which an application developer can define a subset of users, servers or both to provide access to specific database components. A database manager first creates a role and a designer then selects the role for inclusion in an access list for a specific database component.

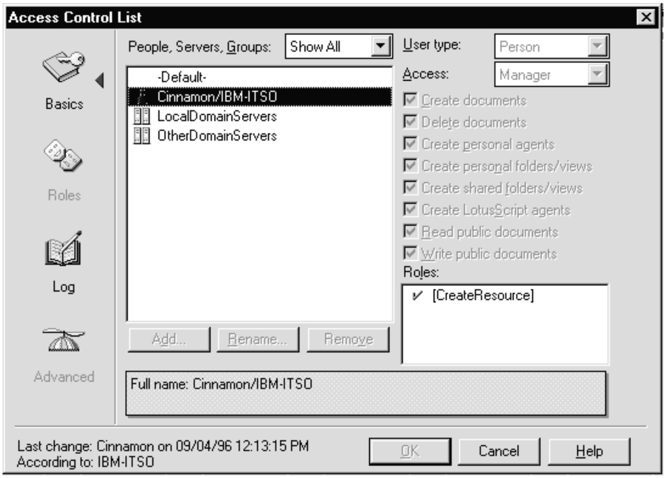
By creating roles within a database ACL and assigning people or groups to it, a Notes developer can restrict or allow access to certain parts of the database or some functionality within the database depending on the current role of the user.

Roles can be thought of as the same as a group within the Name & Address Book, but created and stored at a database level.

Working With Roles

To assign people to a role, you must first create the role in the ACL for the database. To create a role, follow these steps:

1. Select File-Database-Access Control from menu. The Access Control List dialog box is displayed.



Access Control List

People, Servers, Groups: Show All

-Default-
Cinnamon/IBM-ITSO
LocalDomainServers
OtherDomainServers

User type: Person

Access: Manager

☑ Create documents
☑ Delete documents
☑ Create personal agents
☑ Create personal folders/views
☑ Create shared folders/views
☑ Create LotusScript agents
☑ Read public documents
☑ Write public documents

Roles:
✓ [CreateResource]

Full name: Cinnamon/IBM-ITSO

Last change: Cinnamon on 09/04/96 12:13:15 PM
According to: IBM-ITSO

2. Click on the Roles icon to the left-hand side of the dialog box.
3. To add a new role, click the Add button, type in the name of your new role and click OK.
4. To assign this role to a person, click the Basics icon to the left of the dialog box.
5. Select the person or group with the mouse and click the corresponding role in the Roles list in the bottom right of the dialog box. A check mark should appear next to the role meaning that this person now belongs to this role.

Using @Functions in Roles

When creating applications, the application developer can use the @UserRoles() and @IsMember functions to determine whether the current user is a member of a particular role.

For example, when creating applications that are to be published over the Internet, the Notes HTTP server will define people that are using the database from a Web client in the $$WebClient role. You can then easily determine whether to display an HTML formatted document to the user or a Notes document with a formula like this:

@IsMember(“$$WebClient”;@UserRoles);

This formula returns *True* if the person currently logged on is a member of the $$WebClient role.

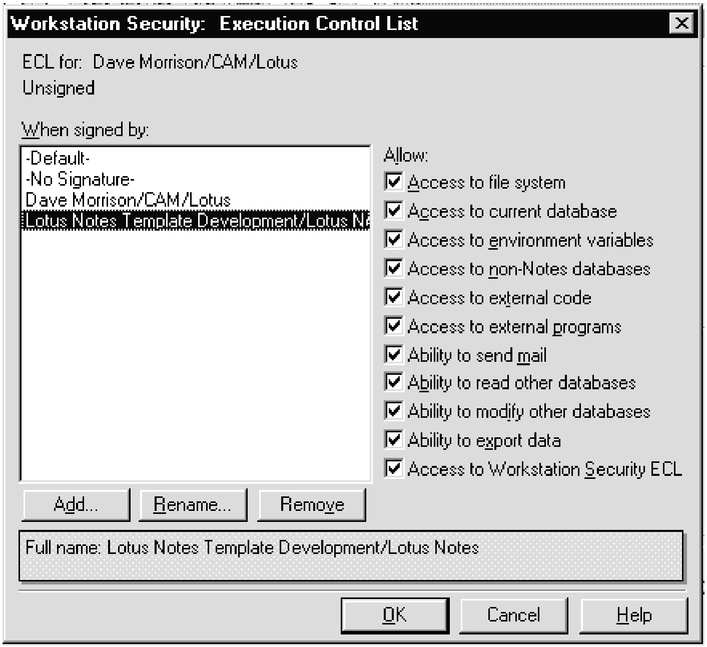
Execution Control Lists

Execution Control Lists (ECL) have stemmed from the concern that a Notes user does not have much control over what a Notes application is doing to their document, database or system.

ECLs are a means by which the Notes user can now specify what level of access an executing formula or LotusScript program created by another person can have to their system.

By default, no scripts or formulas, whether signed or unsigned, can execute on your workstation without displaying a warning message.

ECLs are stored on a user's workstation. To work with ECLs select File - Tools - User Preferences menu. Clicking on the button marked Security Options. The Workstation Security: Execution Control List dialog box is displayed.



Workstation Security: Execution Control List

ECL for: Dave Morrison/CAM/Lotus
Unsigned

When signed by:
- -Default-
- -No Signature-
- Dave Morrison/CAM/Lotus
- Lotus Notes Template Development/Lotus Na

Allow:
- ☑ Access to file system
- ☑ Access to current database
- ☑ Access to environment variables
- ☑ Access to non-Notes databases
- ☑ Access to external code
- ☑ Access to external programs
- ☑ Ability to send mail
- ☑ Ability to read other databases
- ☑ Ability to modify other databases
- ☑ Ability to export data
- ☑ Access to Workstation Security ECL

Full name: Lotus Notes Template Development/Lotus Notes

The dialog is split into two areas. These are the developers signatures and the levels of access that documents signed by these developers can perform on your workstation.

For example, suppose you are trying to create a document from a design that has been signed by the person *Dave Morrison/CAM/Lotus.* You have specified in your ECL list that you do not want to give this person access to your environment variables.

When you open up the document and the program tries to perform an @Environment command, a dialog box will appear telling you that the system is trying to access an environment variable when it does not have the authority to do so. If you want to allow the command to continue you can do so. You can either allow it to run this one time only, or you have the ability to change the ECL permanently to give *Dave Morrison/CAM/Lotus* the ability to access any of your environment variables in the future.

A complete list of LotusScript and @Functions that are affected by the ECL can be found in the Notes Help database.

Note By default every template that comes from Lotus is signed by Lotus Notes Template Development which is given full access to your system.

Document Level Security

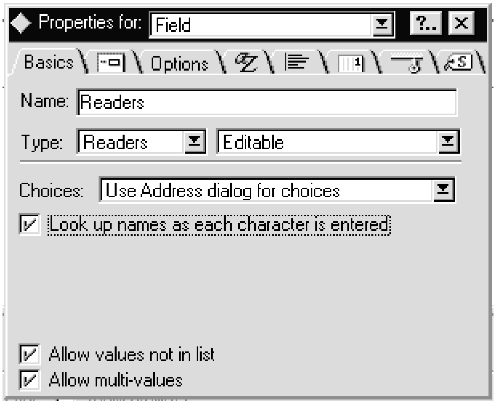
After a user has been given access to a database, it does not necessarily mean that they have access to all the documents within that database.

The first level of security you can implement is at a document level by adding a reader names field to a form.

This special field contains a list of all the people or groups that may read this specific document, no matter what level of access they have been given in the database ACL.

To create a reader field for a form, complete the following steps:

1. Open your form in design mode.
2. Create a new field by selecting Create-Field from the menu.
3. Change the field type to Readers.
4. If you wish the users to be able to add their own list of people to the group, select editable, or provide a formula that will create a list of names automatically.



Properties for: Field

Name: Readers

Type: Readers | Editable

Choices: Use Address dialog for choices

☑ Look up names as each character is entered

☑ Allow values not in list

☑ Allow multi-values

The same follows for users that you want to be able to create documents with a form. Instead of selecting Readers as the field type, select Authors.

A default list of readers and authors can also be created through the forms InfoBox. On the security tab there are two fields that you can add names to for read and create authority with this form. By default, all people with reader or create access to the database in the ACL have access.

Section Level Security

Sections are areas of a document that can be collapsed into a single line. They make navigation in large documents easier as readers can expand a section when they want to read its contents.

Sections can either be Standard or Controlled Access. Controlled Access sections have an Editors List defined either by the application developer at the form level, or by the user at document creation time.

The Editors List in a section gives access to only those people specified the ability to edit fields within that section.

Field Level Security

If document security is essential, you can create an encryption key that will be used to encrypt the contents of all fields marked as encryptable through the Options tab on the field InfoBox.

Creating an Encryption Key

To create an encryption key you need to do the following:

1. Choose File - Tools - User ID from the menu bar.
2. Click the Encryption icon on the left-hand side of the dialog box.
3. Click the Add button, the Add Encryption Key dialog box is displayed.
4. Type in a name and a comment for this key.
5. If the key is to be used by people outside North America select the International option button.
6. Click OK.

Sending the Encryption Key to Other Users

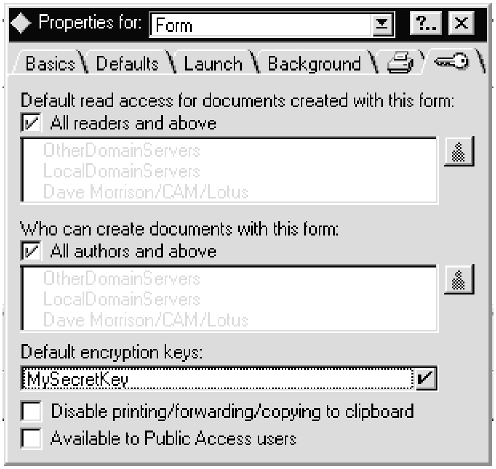
As encryption keys are stored in your ID file, if you want other people to be able to view documents that have been encrypted with this key you must send them a copy of the key first. The easiest way to do this is to select the Mail button from the User ID dialog box. Once the key has been received by the intended user they can merge it into their ID file by choosing Actions - Accept Encryption Key from the menu bar.

Encrypting the Field Contents

Once you have created an encryption key it can be used to encrypt the contents of fields within a document so that others cannot view them.

To enable encryption on a form perform the following:

1. Open the form in design mode.
2. Select Design - Form properties from the menu to display the Form InfoBox.



Properties for: Form

Default read access for documents created with this form:
☑ All readers and above
OtherDomainServers
LocalDomainServers
Dave Morrison/CAM/Lotus

Who can create documents with this form:
☑ All authors and above
OtherDomainServers
LocalDomainServers
Dave Morrison/CAM/Lotus

Default encryption keys:
MySecretKey

□ Disable printing/forwarding/copying to clipboard
□ Available to Public Access users

3. Click the security tab with the key icon.
4. Click on the field marked Default encryption keys. A list of the available encryption keys will appear.

   Note If you have not created any keys, this field will be blank.
5. Select the key you wish to use. A check mark appears next to the key name.

Having set a default key to use with the form, you must now specify which fields are to be encrypted. To do this, perform the following:

6. Open the form in design mode.
7. Either create a new field by choosing Create - Field from the menu or click on an existing field and choose Design - Field Properties from the menu.
8. Click the Options tab on the Field InfoBox.
9. Select Enable Encryption for this field at the bottom of the InfoBox.

When a user now creates a document using this form, the encrypted field you just created will be marked by two red marks at the top left and bottom right of the field. When the form is saved, the encryption key that was specified in the form properties is used to encrypt the contents of this field.

When someone who does not have the correct encryption key opens the document, the field and its contents are hidden.

Part 2
Extending the Reach

Introduction

The second part of this book is dedicated to tools and techniques that enable you to seamlessly integrate an existing IT infrastructure into your Notes applications. As such, the construction of net-aware Notes applications and the integration of external databases play a major role in this second part.

Internet and Intranet Integration

Domino, the integrated Notes / HTTP server, extends the power of your existing Notes applications by publishing them on-the-fly. Domino enables any Web client to participate in Notes applications securely, which enables you to automatically capture and manage information from Web browser users within or outside your company. Hence, your existing Notes applications will become net applications that interact with front-end user interfaces other than the Notes workstation.

The integrated Notes Web server will truly extend the reach of your applications to employees, partners, suppliers and customers over Intranets and the Internet. Even in this environment, it offers you multiple levels of secure access and built-in database integration.

Database Integration

Lotus Notes differs significantly from the design of traditional relational database management systems (DBMS). DBMSs focus on the point-in-time capture of structured data and basic business transactions; Notes focuses on distributed capture of semi-structured data through compound documents. While the systems may seem to be incompatible, Lotus Notes is unique in its ability to complement DBMS applications in such a way that both system types draw upon and reinforce one another’s strengths.

A variety of integration techniques and products are available that allow you to leverage the power of both environments, making Lotus Notes a central access point for all of your enterprise database, network, and Internet resources. In this second part of the document we discuss the integration options that exploit the respective strengths of Notes and traditional DBMSs and transaction systems. They include:

- Native Notes access to DBMSs (LotusScript Data Object, @DB functions)
- Access to Notes databases from DBMSs and query tools (NotesSQL)

- Server-to-server high-volume data transfer (Lotus NotesPump)
- Links to transaction systems (IBM MQSeries link for Lotus Notes)

Each of the available tools for Notes / DBMS Integration has strengths in the areas in which they were designed to excel. These strengths are summarized in the following table:

	LS:DO	***NotesSQL***	***NotesPump***	***MQ Series Link for Lotus Notes***
Data Source(s)	All ODBC-compliant data sources	All ODBC-compliant data sources	DB2, Oracle, Sybase, plus ODBC-compliant data sources	18 host systems via MQSeries systems, e.g., IMS, AS / 400, DEC VMS, CICS
Read & Write?	Yes	Yes	Yes	Yes
Data volatility?	High	High	Low	High
Response time?	Real-time or batch	Real-time	Batch and event	Real-time and asynchronous
Programming involved?	Yes	Depends on application used	Not required; available	Yes
Volume of data transfer?	Moderate	Moderate	High	Moderate
Notes client / server support?	Client and Server	Client and Server	Server	Client and Server

Chapter 13
Domino: Architecture and Configuration

Overview

This chapter describes the Domino technology in Lotus Notes based on Domino 1.5. The Domino architecture is introduced and its elements are discussed.

The main areas covered in this chapter are:

- The Internet and the World Wide Web
- Lotus Notes and the Web
- Domino Architecture
- Configuring the Domino Web server
- Accessing a Domino site.

The Internet and the World Wide Web: An Introduction

The Internet is a worldwide collection of computers organized into a series of networks. This wide-area network (WAN) collection of networks includes the ARPANet, NSFnet, regional networks, local networks at universities and research institutions, and a number of military networks. The term “Internet” refers to this entire set of networks.

It evolved from developments in the scientific and defense community, where there was a need for scientists to share and collaborate on research projects. This requirement led to a government funded network of these scientific networks and computers connected through the TCP/IP protocol which became known as the Internet. The Internet provides the following main functions:

- Correspondence through electronic mail through the Simple Mail Transfer Protocol (SMTP)
- Information discussion and sharing through USENET Newsgroups using NNTP

- File transfer enabled by the File Transfer Protocol (FTP)
- Terminal Emulation and remote login through Telnet
- Compound Document Publishing through the World Wide Web (WWW) and Hypertext Transfer Protocol (HTTP)

Although the Internet has been around for almost two decades, it is only in the mid 1990's that its use has become widespread. This is primarily because through commercialization of the Web, more funding has allowed better network bandwidth and overall Web content.

The number of people using the Internet grows daily with current 1996 estimates at between 40 to 60 million. The Internet population is growing exponentially as business tries to use this electronic venue as the new way to market its products and services.

The World Wide Web

Clearly the World Wide Web provides the most functionality of all the Internet services.

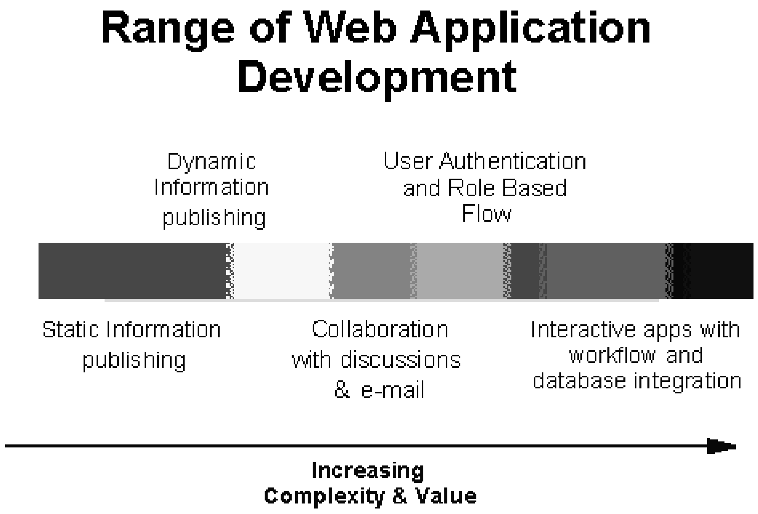
Its main strength is the ability to present documents, with text, graphics, audio, video, hypertext links to other documents, and other compound multimedia content to any Web user worldwide. These documents are commonly called Web pages and are presented to Web users through HTTP over the Internet's transport protocol TCP/IP. These Web pages are written in HTML code which describes the information and the presentation delivery of the documents. They are accessed by way of reference through a Universal Resource Locator (URL) that is requested through an HTTP command.

Browsers

Programs that allow users to quickly and easily interface with Web documents are called browsers. They provide a graphical user interface to display Web documents and make HTTP requests.

Companies are not only advertising their products and services on the Web, but creating applications that are key to their business processes. Information, products, and services can now be procured through the Web and the process by which this is done is increasingly an electronic one.

A Web page has the possibility to be more than static text. The Domino Web server technology for Lotus Notes enables you to bring all these functions to your Web applications.


Range of Web Application Development
Dynamic Information publishing
User Authentication and Role Based Flow
Static Information publishing
Collaboration with discussions & e-mail
Interactive apps with workflow and database integration
Increasing Complexity & Value

There is another benefit to the use of the Internet and Web. You can use its connectivity and bandwidth to connect your different offices, people and customers by building loosely coupled business environments and applications that are connected through the Internet or your own private network. These applications are called intranets and they can range from publishing information between company locations to more sophisticated corporate and departmental applications that support business processes over the Internet's network and between possibly disparate technologies.

This requires adding functionality to the Web to allow transaction-oriented applications, workflow, mobile use, and collaboration all through secure, private Web experiences.

Lotus Notes technology is proven in the workgroup environment with thousands of companies using it for key business applications. The Web is another environment where Notes can bring its functionality to bear greater benefits for customers.

Internet and Web Terminology

Before we begin to discuss Domino, there are a number of terms you need to understand. The following terms will be discussed at more length in this chapter.

HTML	Hypertext Markup Language; the document format for the Web.
WWW	World Wide Web
HTTP	Hypertext Transfer Protocol is the standard Internet protocol that enables Web clients to talk to Web servers.
URL	Uniform Resource Locator designates the unique location of a site / page.
CGI	Common Gateway Interface
NNTP	Network News Transfer Protocol is the protocol used by UseNet newsgroups.
SMTP	Simple Mail Transfer Protocol is a protocol used to send text-based E-Mail.
MIME	Multi-Media Internet Mail Extensions are an enhancement to the SMTP protocol that enables the transfer of binary files.

Lotus Notes and the Web

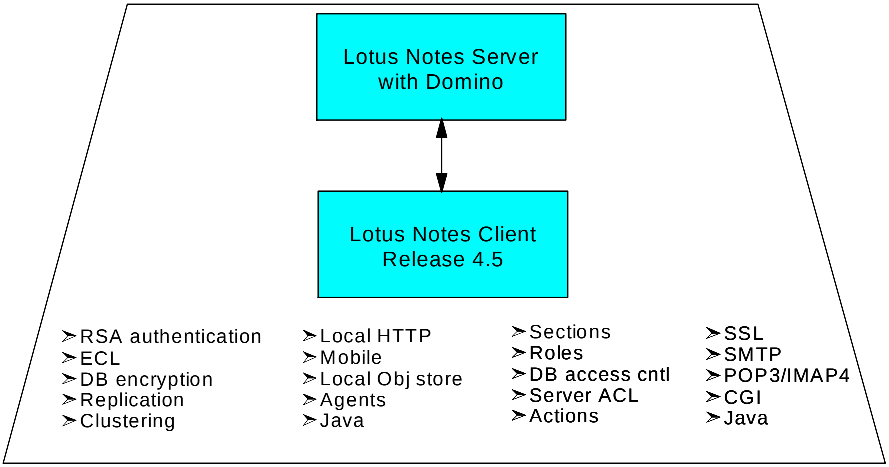
Lotus Notes is ideally suited for Web and intranet environments. With the Domino technology, it goes beyond today’s HTTP server to bring the Web a rich set of proven functions from the Lotus Notes workgroup world.

Domino is server technology that transforms Lotus Notes into 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.

Beyond static information dissemination and browsing, Notes has robust and industry-leading technology in:

- Programmable object services
- Information push and pull replication
- Messaging / directory services
- Database integration
- Transaction system integration
- Security and RSA authentication
- Workflow, tracking, collaboration, and conferencing

- Search engines
- Document management and linking
- Scripting and agent support
- Network and mobile support
- Administration
- HTTP, HTML, CGI, and Java support



About the Lotus Notes Client Web Navigator Feature

The Web navigator is a feature of Notes that provides Notes users with easy access to information on the Internet. The Web navigator combines the features of a Web browser with all the groupware capabilities of Notes to allow Notes users the ability to access information on the Internet from the familiar surroundings of their Notes environment.

With Notes and the Web navigator, you can:

- Browse the Web on your own using the client-side retrieval capabilities of the Personal Web navigator
- Browse the Web with others using the server-based retrieval capabilities of the Server Web navigator (without having to install and configure TCP/IP on each user's workstation)
- Browse through previously-retrieved Web pages while disconnected from the Internet
- Save Web pages in any Notes database for future reference

- Share Web pages with coworkers in a shared database
- Capture Web pages within any Notes application, such as a Customer Tracking or Competitor Watch database
- Program Web access and navigation features into any Notes application
- Rate, annotate, and categorize Web pages
- Use the built-in intelligent agents to locate and return Web pages of interest to the user, or create your own customized agents
- Access active Web pages (pages that contain Java applets, plug-ins, and helper applications)
- Create a customized browser application
- Browse the Web and save your browsing experience as a Web Tour that can be reloaded at a later time
- Gain one-click access to a preferred search site.

About the Domino Architecture

Domino is a Lotus Notes server task. It enables a Lotus Notes server to become an HTTP server. Domino can be thought of as:

- A Web server
- A Web application development environment
- An enabler of an alternate client for Notes: the Web client.

Domino integrates the open networking environment of Internet standards and protocols and the powerful application development facilities of Notes. This environment provides you with the ability to rapidly develop a broad range of business applications for the Internet and the intranet.

The Domino Web server merges Web server technology with Notes technology to allow any Web browser to access data and applications stored in Notes databases. Web site designers can use Domino to build applications that take advantage of core Notes functionality, such as replication, full-text search, application development, security, and workflow.

Domino is a Web server providing logging, configuration, and security management features as well as on-the-fly, dynamic integration of Notes and Web environments. It speaks the HTTP protocol, converting HTML code on the fly, thus enabling Web clients to communicate with Notes servers and service the requests. Through a URL interface, Domino examines incoming HTTP requests and responds in one of two ways:

- If the request is for an HTML file in the file system, Domino acts like an HTTP server displaying HTML documents.
- If the request is for a Notes database element, the Domino engine interacts with the Notes database to retrieve the appropriate information to return to the requesting HTTP process or save information in the database.

Notes views, forms, navigators, and links are translated into HTML on the fly for display on any Web client. This allows you to instantly take your Notes applications and serve them over Domino. Domino basically exposes Notes design elements as an extension of the URL interface by attaching the Notes element and a command on the end of the HTTP service request to access the Notes element. For example:

```
http://www.millenia.com/domino.nsf?OpenDatabase
```

See the URL Syntax section in this chapter for more details.

Configuring the Domino Web Server

Configuring the Domino Web server involves the following main areas:

Setting Up Your Notes Server on the Internet

System Requirements

Domino has the same system requirements as your Notes server. The Domino files do not occupy much disk space; the hardware requirements include disk space allocated for the file cache and log files. For example, on NT:

- NT Server 3.51
- 1 gigabyte disk drive
- 64MB of RAM

Network Requirements

The PC on which you install Domino should have the following network connectivity:

- A connection to a company LAN or intranet that uses TCP/IP as a protocol
- TCP/IP on the Notes server where the Domino software resides

Domino supports the following TCP/IP implementations:

- Windows NT server™ — TCP/IP that comes with the Windows NT software

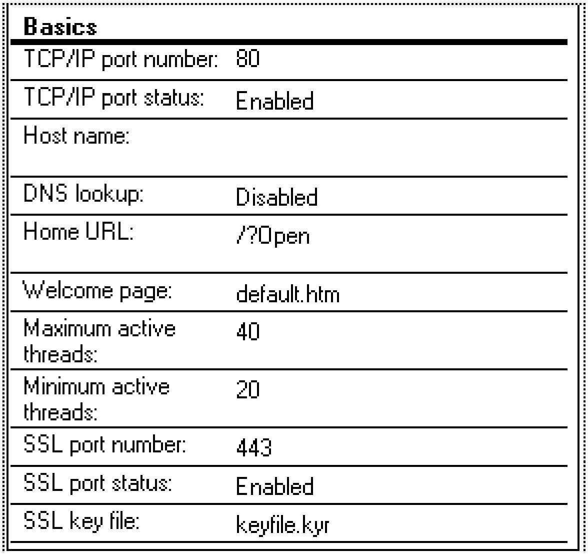
If you plan to use Domino to manage an external Web site, you'll need:

- An Internet connection through a leased-line or dial-up connection to an Internet Service Provider (ISP).

Connecting to the Internet raises potential security risks. Create and maintain a secure environment by installing a firewall or creating separate internal and external networks.

HTTP Setup

Configuration settings for the Domino Web Server are stored in the HTTP Server settings section of the Server document in your server's Name and Address Book.



Basics

TCP/IP port number:	80
TCP/IP port status:	Enabled
Host name:	
DNS lookup:	Disabled
Home URL:	/?Open
Welcome page:	default.htm
Maximum active threads:	40
Minimum active threads:	20
SSL port number:	443
SSL port status:	Enabled
SSL key file:	keyfile.kyr

TCP/IP port number (default=80)
Specify the port number on which you want the Domino server to listen for HTTP requests. The industry standard port number for HTTP is 80. Common ports used are 8080 and 8008.

Note Do not use port numbers less than 1024 (except for the default of 80), which are reserved for other TCP/IP applications.

TCP/IP port status (default=Enabled)
Specify the status of the TCP/IP port. Either the TCP/IP port or the SSL port must be enabled for Domino to operate.

Host name (default=blank)
Enter the fully qualified host name that is returned to the browser. If your PC does not have a host name registered in a DNS, enter the PC's IP address in this field.

DNS Lookup (default=Disabled)
Specify whether you want the Domino server to look up the DNS host name of the requesting client.

Tip If you enable DNS, your server works harder to perform host name lookups. This also causes storage of long host names for log file and log filter entries.

Home URL (default=/?Open)
Specify the URL you want Domino to return when users enter a site name directly, but do not specify an explicit directory or page name (for example, http://domino.lotus.com).

Using the default setting, /?Open, Domino displays a list of databases on the server. This is equivalent to the File - Database - Open command in Notes. To have Domino look for and return the Welcome page in the HTML directory, leave this field blank or specify /default.htm.

Specifying a URL that begins with a / (slash) causes Domino to return the URL information directly to the browser. The browser still displays http://hostname.domain.com/ in the location text box.

Specifying URLs that start with a protocol such as http://host.domain.com/ causes Domino to send a redirected URL to the browser. That is, the browser performs an HTTP GET request on the specified URL. The information in the browser's location text box then changes to what is specified in this field.

Examples: /domino.nsf

```
/dominodisc.nsf/By+Author
/dominodisc.nsf/$defaultnav
http://myhost.domain.com/home/myhome.html
```

Welcome page (default=default.htm)
Specify the default page file name you want the Domino server to load when a client accesses a directory not followed by an explicit page name.

Maximum active threads (default=40)
Specify the maximum number of threads you want to have active at one time. If the maximum is reached, the Domino server holds new requests until another request finishes and threads become available. The more

power your PC has, the higher value you should use. If your PC spends too much time on overhead tasks, such as swapping memory, reduce this value.

Minimum active threads (default=20)
Specify the minimum number of threads you want the Domino server to use or have available to use. The Domino server will not close threads below this minimum even if the threads are idle. The more power your machine has, the higher value you should use. If your PC spends too much time on overhead tasks, such as swapping memory, reduce this value.

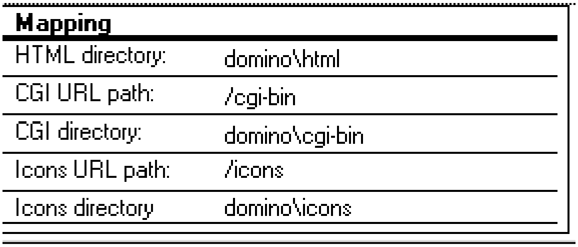
SSL port number (default=443)
Specify the port for SSL security. The Domino server uses this port only for HTTP requests. Requests for HTTP will still come on the port that you set with the TCP/IP port. If you change this setting, you must stop the Domino server and restart it so the changes take effect.

SSL port status (default=Enabled)
Specify the status of the SSL port. Either the SSL port or the TCP/IP port must be enabled for Domino to operate.

SSL key file (default=keyfile.kyr)
Specify the name for the key file. If you change this setting, you must stop the Domino server and restart it so the changes take effect. The key file is stored in the Notes data directory by default. To store the key file in a different directory, specify a full path.

Mapping settings

The mapping settings are stored in the HTTP Server settings section of the Server document in your server Name and Address Book and direct Domino where to look for each of its component files.



Mapping

HTML directory:	domino\html
CGI URL path:	/cgi-bin
CGI directory:	domino\cgi-bin
Icons URL path:	/icons
Icons directory	domino\icons

HTML directory (default=domino\html)
Specify the directory location for HTML files. The directory is relative to the Notes data directory unless a full path is specified.

CGI URL path (default= /cgi-bin)
Specify the URL path to the CGI programs directory. Note that this path relates to URLs and not the file system.

CGI directory (default=domino\cgi-bin)
Specify the directory location for CGI program files. The directory is relative to the Notes data directory unless a full path is specified.

Icon URL path (default=/icons)
Specify the URL path to the Domino icons directory. Note that this path relates to URLs and not the file system. In general, you do not need to modify the icons fields. However, if you have an existing icons directory, specify the path to the directory here.

Icons directory (default=domino\icons)
Specify the directory location for the icons directory. The directory is relative to the Notes data directory unless a full path is specified.

Operation Information Settings

Operational information settings stored in the HTTP Server settings section of the Server document in your server Name and Address Book.

Cache directory (default=domino\cache)
Specify the directory for the Domino server to use as a cache. Domino uses this directory to store graphic images stored as GIF files and file attachments.

Maximum cache size (default=50 MB)
Specify the maximum amount of available disk space, in megabytes (MB), you want the cache to use.

Delete cache on shutdown (default=Disabled)
Specify whether you want Domino to delete the cache when you shut down the server.

Garbage collection (default=Enabled)
If you have enabled caching, the Domino server uses the garbage collection process to delete files that should no longer be cached from the least to the most frequently accessed, which means that the most frequently accessed files are the last files Domino deletes.

Garbage collection interval (default=60 minutes)
Specify a time interval, in minutes, at which to run the garbage collection process.

Image conversion format (default=GIF)
Specify the image file format you want Domino to use when converting image files. The options are GIF and JPEG.

Interlaced rendering (default=Enabled)
If you chose GIF as the image conversion format, specify whether or not you want Domino to render the GIF images in an interlaced format.

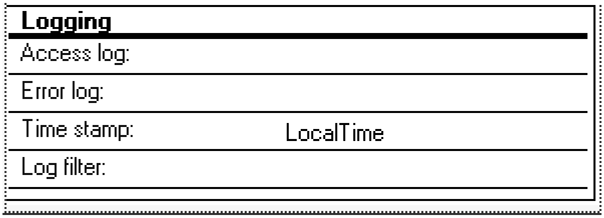
Progressive rendering (default=Enabled)
If you chose JPEG as the image conversion format, specify whether or not you want Domino to render the JPEG images in a progressive format.

JPEG image quality (default=75)
If you chose JPEG as the image conversion format, specify the percentage numeric value for the JPEG image quality. The range is 5 to 100 percent. The larger the value, the larger the file, and the better the image quality. The lower the value, the smaller the file, the less time it takes to transmit, and the lower the image quality.

Default lines per view (default=30)
Specify the default number of lines Domino uses to display a Notes view. Note that this setting affects every database on the Domino server.

Logging settings

Domino creates an access log and an error log. Both of these are specified in the Logging section of the server document. The files created are relative to the Notes Data directory with a timestamp suffix. A log filter can help you decrease the size of your log files.



Logging

Access log:	
Error log:	
Time stamp:	LocalTime
Log filter:	

Access log (default=blank)
Specify the path and / or the file name where you want the Domino server to log access statistics.

Error log (default=blank)
Specify the path and the file name where you want the Domino server to log internal errors. Note that the path is relative to the Notes data directory.

Time stamp (default=LocalTime)
Specify whether the log files should record entries using local time or Greenwich Mean Time (GMT).

Log filter (default=blank)
Use this option to specify host names or domains whose access requests you do not want to log. You may want to suppress log entries for certain hosts or domains. Specify the IP number or host name template in this field.

Examples: NoLog template

```
NoLog 128.141.*.*
NoLog *.cern.ch
NoLog *.ch  *.fr  *.it
```

To assign the same setting to template names, separate them by one or more spaces.

Note To use host name templates, you must enable the DNS Lookup setting. If the DNS Lookup option is disabled, you can use IP address templates only.

```
Example:  www.internotes.lotus.com; www.lotus.com;
www.ibm.com
```

Registering Web Users

You must set up user authentication at the server by creating Person documents and adding HTTP passwords for all Web users who are allowed to access the Domino server and applications.

Creating a Person Document for a Web User

If you manually create Person documents for Web users in the Public Address Book, the minimum information required is a user name in the "User Name" field and a password in the HTTP Password field. When you save the Person document, the HTTP password is encrypted.

A Registration Application

The Domino Web site, http://domino.lotus.com, has a sample registration application available for you to download. The sample registration application is described in the chapter on Domino sample applications.

Starting and Stopping the Web Server

Starting the Server Manually

1. Start the Notes server.
2. At the console, enter the command:

   ```
   load http
   ```

Starting the Server Automatically

To start Domino automatically whenever the Notes server starts, bring the server down and do the following:

1. In the NOTES.INI file, add the *http command* to the line that begins with ServerTasks=
2. Save the file and restart the server so the changes take effect.

Stopping the Server

To stop Domino, enter this command at the console:

```
tell http quit
```

Setting Up Security

Domino leverages the Notes access control model to control access to Web applications built on it.

About Securing Access to Your Environment

The following features allow the Domino-enabled application to provide superior secure access controls:

- Basic Web user authentication through the Notes Name and Address Book.
- User registration management through a pre-built template.
- Roles-based security.
- HTTP request security by activating the Secure Sockets Layer (SSL).
- Notes ACL security for individual databases and in finer granularity for Notes elements like views, forms, or fields.
- Directory browsing control set up in the Name and Address Book.

Note The Domino Web Server does not allow “passthru” to other Notes servers. Web users are restricted to accessing databases only on the Domino Web server.

About Domino User Authentication

Just as the Notes ID file is the foundation for security on a Notes server, the Web user name and password provide the entrance to the Web site’s databases and are used as the basis for determining what an individual can accomplish at the site. The technique by which users are granted access to the Domino Web server is known as basic Web authentication. This established standard for Web security is based on a challenge/response protocol.

Web users are only authenticated when they attempt to do something for which access is restricted. For example, when users try to open a database whose default access is No Access, they are challenged by the server to supply a valid name and password. Authentication succeeds if the user name and password supplied match the appropriate fields in the Person document of the Public Name and Address Book on the Domino Web server.

Caution Basic Web authentication is not considered as secure as Notes public key certificate-based authentication because it doesn't require User IDs to validate the client's identity.

Opening Access to Anonymous Web Users

Any unregistered Web user who tries to access a Domino Web server without a valid user name and password is known by the name "anonymous." You need to decide what level of database access you want anonymous users to have.

To make your databases widely available to unregistered Web users, create an entry in each database's Access Control List called "Anonymous" and assign it the appropriate access such as Reader.

Caution If there is no entry for "Anonymous," such users receive the default access set in the database ACL.

Tip To protect the databases from unregistered Web users, create an entry in each database's Access Control List called "Anonymous" and grant it "No Access."

About Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is a security protocol that provides communications privacy and authentication over the Internet. The Domino Web server can be configured to encrypt data as it passes between Web clients and the server.

Benefits of SSL Transactions

- Privacy – HTTP data is encrypted to and from clients, so privacy is ensured during transactions.
- Message validation – An encoded message digest accompanies data to detect any message tampering.
- Server authentication – The server's digital signature accompanies messages to assure the client that the server's identity is authentic.

About Certification Authorities and Key Ring Files

SSL encryption uses the public / private RSA-based crypto system to encrypt data. This system requires the server to hold a unique pair of mathematically-related keys — a private key and a public key — that are used to initiate SSL-encrypted transactions.

Key ring files store the information needed for encrypted transactions: the owner's private and public key and one or more certificates. The encryption process that occurs between the Domino Web server and a Web client is based on the relationship between the key pairs and the certificates.

The link that allows a server and a client to communicate is a Certification Authority (CA). Like a mutual friend, a CA vouches for the identity of a server and client. A CA can be an external, commercial certifier, such as VeriSign™, or an internal certifier that you create at your company. When a CA issues a certificate (known by VeriSign as a Digital ID℠) to a server or client, it includes a trusted root key that contains its name and public key. An SSL transaction is authenticated only if the client can verify the server's identity (because the server has a trusted root key from the same CA).

About Database Access Control Lists

Every database has an access control list (ACL) that defines who can access a database and what tasks users can perform. Users or servers with Notes IDs, or any registered Web users (those whose names and passwords are stored in the Public Name and Address Book), can be listed in the ACL. The manager of each database on the Domino Web server should define an access control list (by choosing File - Database - Access Control) that matches the confidentiality requirements for the information.

Registered Web users can be listed individually, or as members of groups or roles. The most frequently used access levels for ACLs on an interactive Web site are:

- **No Access** Users assigned No Access cannot access the database.
- **Depositor** Users assigned Depositor access can create documents but can't see any documents in the database views, even the documents they create. This type of access is useful for survey responses or ballot boxes.
- **Reader** Users assigned Reader access can read documents in a database but cannot create or edit documents.
- **Author** Users assigned Author access can create documents and edit documents they create.
- **Editor** Users assigned Editor access can create documents and edit all documents, including those created by others.
- **Designer** Users assigned Designer access can create and modify design elements within the database, create a full-text index, and can modify replication formulas.

- **Manager** Users with Manager access can modify ACL settings and also perform all tasks allowed by other access levels (these tasks must be done from a Notes workstation). Notes requires at least one Manager for a database, but it's best to assign two people Manager access.

To further refine a user's or group's access level, you can allow or restrict these specific tasks. For most databases, you should allow Web users to create and delete documents. Other tasks, such as creating folders and views and creating agents, do not apply to Web users. For further explanations on ACL levels, please refer to the chapter on Security in this book.

Common Web Application ACL Settings

- **High-security confidential database**

• Default	No access
• Anonymous	No access
• Authorized readers	Reader
• Contributors	Author
• Supervisors	Editor

- **Medium-security project database**

• Default	Reader
• Anonymous	No access
• Project team members	Editor

- **Low-security informational database**

• Default	Reader
• Anonymous	Reader
• Contributors	Author
• Supervisors	Editor

- **Low-security discussion database**

• Default	Author
• Anonymous	Author
• Supervisors	Editor

Securing Notes Design Elements

You can apply security features into views, forms, and fields to further refine who can see or change specific information on the Web site.

By placing an Anonymous entry in the ACL with "No access," you cause Domino to prompt for the user name and password of the Web user. When logged in, the user has the rights applied to the specific design elements.

Customizing views and folders

Read and edit access control lists (chosen from the Security tab of the View Properties InfoBox) can be useful for organizing information for specific types of users.

Note @UserName does not work in view selection and view column formulas as they are not computed dynamically by the indexer.

Customizing Document-Level Security

A document can restrict which users can read its contents, even those with Reader access to the database. Reader restrictions can be applied to a document in any of these ways:

- Form's read access list (in the Security tab of the form properties InfoBox)
- Readers listed in a Reader Names field on the form
- Documents read access list created by the author or editor in the document properties InfoBox
- Create access lists associated with a form restrict which Web users can open documents created with a certain form.

Customizing Field-Level Security

- Author fields expand the list of authorized editors for specific documents. An Author's field allows you to expand a document's editing privileges to users who didn't create the document, without giving those users Editor access in the ACL.
- Reader fields limit who can read specific documents. You can use this feature to show different users different collections of documents, even though they are actually using the same view to see them.
- Editor-only fields include the field security option, "Must have at least Editor access to use." You can use this feature to restrict authors from editing part of their own documents.

Caution Do not rely on encrypted fields if Web users are authorized to read documents that contain encrypted fields. Field-level encryption does not work for Web users.

Hide When Access

@UserName makes the Web user's name available to Notes formulas. "Hide-when" formulas can be used to hide portions of documents based on the user name.

@UserRoles can be used to check whether the user is a Web client or a Notes client and thereby hide portions based on the user role.

Some More Details on Security With Domino

Local database encryption, mail encryption, document encryption, and network transaction encryption are not functional over Domino. However, SSL encryption is available.

Server access lists which control Notes server activities are not functional over Domino.

ACL control in databases reached through directory pointer (.DIR) files is not functional over Domino.

Access-controlled sections restricting access to sections within a document do not work with databases on a Domino Web server.

Signatures normally available for collapsible sections and mail-enabled documents are not functional.

Domino Log and Cache

About the Domino Log Files

Domino creates log files to log statistics on the use of the Domino server. Domino starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating a file, Domino uses the file name you specify and appends a date suffix. The date suffix is in the format *mmmddyy*, where *mmm* is the first three letters of the month, *dd* is the day of the month, and *yy* is the last two digits of the year. For example, agent_log.jul2196.

The Domino log files are:

Access Log
Domino logs access statistics in this file. By default, Domino writes an entry to this log each time a client sends the server a request unless filtered through the Domino filter. For example:

```
165.238.196.55 - - [21/Jul/1996:00:00:20 +0500]

"GET
/DomGuideApp.nsf/127a8ca2ba2b99bb85256362000651fc/57e48c81b1d
0d72b8525636300" 302 407

165.238.196.55 - - [21/Jul/1996:00:00:22 +0500]

"GET
/DomGuideApp.nsf/127a8ca2ba2b99bb85256362000651fc/11d19a41400
9926c8525636300" 200 1558

165.238.196.55 - - [21/Jul/1996:00:00:35 +0500]
```

Agent Log

Domino logs the type of Web client accessing your site (for example, Netscape, Mosaic, etc.). Here's some sample data from an Agent log:

```
[21/Jul/1996:00:00:20 +0500] "Mozilla/2.0 (compatible; MSIE
3.0B; Windows 95)"
[21/Jul/1996:00:09:31 +0500] "NetCruiser/V2.1.1"
[21/Jul/1996:00:10:05 +0500] "Lotus-Notes/4.1 (OS/2 Server)"
```

Error Log

Domino logs errors to this file. Here is an example:

```
[21/Jul/1996:00:01:41 +0500] [OK] [host: 206.104.23.192
referer: http://domino.lotus.com/

domino.nsf/4432de3b22c1708785256316007db7a0/4262ad9ea85ecc998
525636c008029cf?OpenDocument]

/dominorel.nsf/6e8b6c6ac4e689e085256324006d4ddc/9271311eaf15d
5398525636300529e28

[21/Jul/1996:00:01:52 +0500] [OK] [host: 206.104.23.192
referer:
```

CGI Error Log

Domino writes standard error (stderr) from CGI programs to the CGI error log. Domino creates the CGI error log file on the same path you specify for the error log file and names the file cgi_error.

Referer Log

Domino logs the URLs that clients visited and that contained links to URLs on this site. Here's some sample data from a Referer log:

```
[21/Jul/1996:00:01:07 +0500]"http://domino.lotus.com/"

[21/Jul/1996:00:01:07 +0500]"http://domino.lotus.com/"

[21/Jul/1996:00:01:08 +0500]"http://domino.lotus.com/"

[21/Jul/1996:00:01:09
+0500]"http://domino.lotus.com/domdown.nsf/Software/37f67470d
4dd2b288525631c00732b16?Op"

[21/Jul/1996:00:01:11 +0500]"http://domino.lotus.com/"
```

About the Domino File Cache

Domino uses a file cache directory to optimize response time. Domino stores image files and file attachments in the file cache directory. Since converting bitmap files to inline GIF files can take time, caching the converted files on disk allows Domino to return inline images more quickly. Similarly, file attachments are usually compressed in Notes; storing attachments in the cache improves server responsiveness.

The Domino file cache directory is named domino\cache; its location is relative to the Notes data directory.

The format of the cache file names is:

```
<note-unid>.<field-name>.<item-name>.<item-id>.<item-offset>.
<filetype>
```

For example, an inline graphic might be named:

```
8dbf0dbdd7c002d1852561a300722f42.Body.0.a8e.gif
```

Note The cache file format information is subject to change.

Accessing a Domino Site

Creating, Editing, and Deleting Documents From the Web

Creating Documents From the Web

When you access a Domino site and click a link to a form, Domino loads the form “on-the-fly.” In other words, Domino calculates formulas the form uses, such as author name, date/time created, date/time last modified, or default values based on user name — before displaying the form to you. When you submit the form, Domino creates a document in a Notes database. Once the information is in Notes, you can run an agent, mail the document, trigger a workflow process, and so on.

Editing Documents From the Web

To edit a document, open it, click the EDIT button, and make your changes. Click the SUBMIT button to have Domino save the edited document in the Notes database.

You can edit any document you wrote or, if your access level is Editor or higher, you can edit other people’s documents provided the @Command([EditDocument]) action has been put behind the appropriate action bar button. For example, look at a question you didn’t write in the

Domino Discussion on the Domino Web site; only the NEW and RESPOND buttons are available. Look at a question for which you are the author; the EDIT and DELETE buttons are available as shown below:

Deleting Documents From the Web

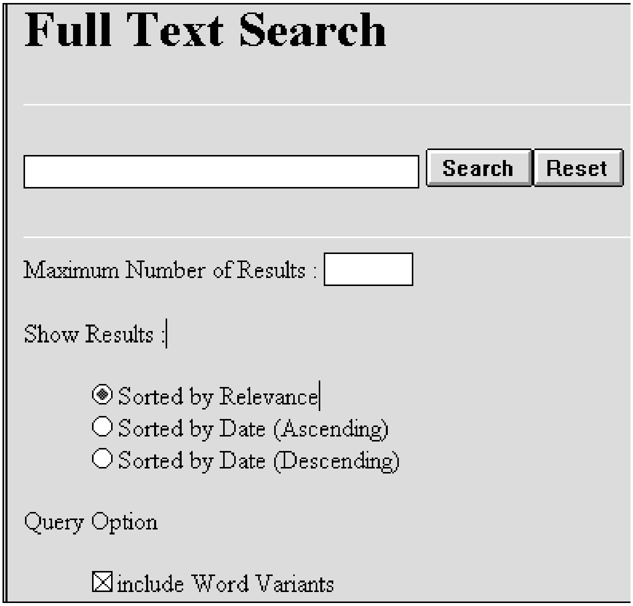
To delete a document, you must first open it because there is no concept of a “selected document in a view” on the Web. You can delete any document for which you are the author or editor.

Note Actions that depend on the document, such as edit, delete, or respond, must be included as form actions.

Searching a Domino Site

Search Options Dialog Box

The Full Text Search dialog box lets you specify your search and how the search results appear.



Full Text Search

Maximum Number of Results :

Show Results :

● Sorted by Relevance
○ Sorted by Date (Ascending)
○ Sorted by Date (Descending)

Query Option

☒ include Word Variants

Viewing Search Results on the Web

Search results appear as a list of hypertext links to documents. Domino displays search results in the format of the view in which the search was performed.

Along with the search results, Domino displays a search bar that allows you to refine the current search or enter a new one.

Note Documents added to the database since the database's full text index was last updated will not be returned by the search.

Reading and Responding to Notes Mail

When your mail database resides on a Domino server, you can read and send mail over the Internet, giving you access to a Notes Mail account from a Web browser. Domino gives you the flexibility of choosing to use Notes Mail from any workstation connected to the Internet or from any Web browser. Even if you don't have your Notes ID with you, you can read and send mail by entering the URL for your mail database and supplying your name and password. Certain commands that are available from a Notes workstation cannot be converted for the Web. For example, there is no "OK/Cancel" confirmation dialog; instead, mail is sent or deleted without a confirmation dialog.

Be aware of the following issues when accessing mail from a Web client:

- Even if your Domino Web server is set up for Secure Sockets Layer (SSL) transactions that protect privacy and authenticate Web users, you should know that SSL offers less security than the Notes ID security system that is activated when accessing mail from a Notes workstation.
- You cannot store mail in a local replica.
- Send options are more limited than those available at a Notes workstation (for example, the Forward option is not available).

Preparing to Access Notes Mail From a Web Client

Before you can use Notes Mail from a Web client, the Domino administrator must:

- Register you as a Web user in the Public Address Book
- Create a replica of your mail database on the Domino server or add Domino to the mail server

Chapter 14
Domino: Creating Web Applications

Setting Up Your Web Site

This chapter builds on the content of the previous chapter and describes how to use the Domino technology within Lotus Notes.

The main areas covered in this chapter are:

- Introduction to Lotus Internet Applications
- Web Site Organization
- Web Application Design Elements
- Adding HTML to Notes Elements
- Creating Links

This chapter was written using Release 1.0 of the Domino server.

About Web Applications

Once your Domino server is set up, you can begin to design your application elements for a system to be used over the Web. The main areas to consider are the following:

- Web Site Organization
- Web Application Design Elements
- Adding HTML to Notes Elements
- Creating Links

With Domino, the Lotus Notes application development environment is composed of the following rich functionality:

- Delivery of **dynamic** content based on:
  - Time
  - User input
  - Hide When Formulas
  - User identity
  - Web or Notes client type

- Collaborative applications such as threaded discussions.
- RDBMS systems access.
- Streamlined and automated business processes with workflow applications.
- Secure Web applications using Access Control Lists and roles for granular Notes element access to:
  - Databases, Views, Documents, Forms, Fields
- Directory services through the Notes Public Name Address Book (NAB), for managing Web users as individuals or in user groups.
- Basic Web Authentication.
- Secure Sockets Layer (SSL) support for server authentication and encryption of data in secured sessions.
- Full CGI application support.
- MIME type mapping of data and file objects stored on the server.
- Java script in-line support.
- Java applications can be referenced from any page on the Web site from either the file system or the Notes object store. As a result, any Java development environment can be used in conjunction with Domino.

Domino can be used in conjunction with streaming audio and video servers to serve plug-ins like Shockwave.

Introduction to Lotus Internet Applications

Domino provides the platform for Lotus' Internet Applications, which are business solutions enabling the deployment of Internet and Intranet technology in the organization.

These applications build on the strengths of the Domino environment — its rapid application development environment, object services, robust security, easily administered replication of data, and support for multiple operating systems — to efficiently integrate legacy systems, provide the platform for authoring and approval workflows required to create content, and manage the interactive communications — the discussions, the surveys, the transaction-processing applications — that make this new world of standards-based communication so exciting.

Lotus' Internet Applications create a maximum 'net presence with minimum effort. The SiteCreator that comes with each Internet Application is designed to provide a "0 to Web site" implementation path quickly and effectively, without extraordinary efforts and cost. Yet the out-of-the-box usefulness of the Internet Applications doesn't imply any reduction in Notes' long-standing emphasis on end-user development.

The Domino server and Lotus' Internet Applications allow developers to customize the applications. Domino adds Notes' programmable workflow and security, the replication model for supporting distributed authoring, the built-in interactivity of a threaded discussion model, and the ability to process intelligent forms and store the results as actionable fielded data in a database, rather than just as text.

Because organizations can develop applications that reach both the Notes client and the Web browser, they gain the freedom to choose the best delivery solution for the applications and the audience.

Domino.Action

The starting point for any organization is the establishment of an electronic presence on the Web.

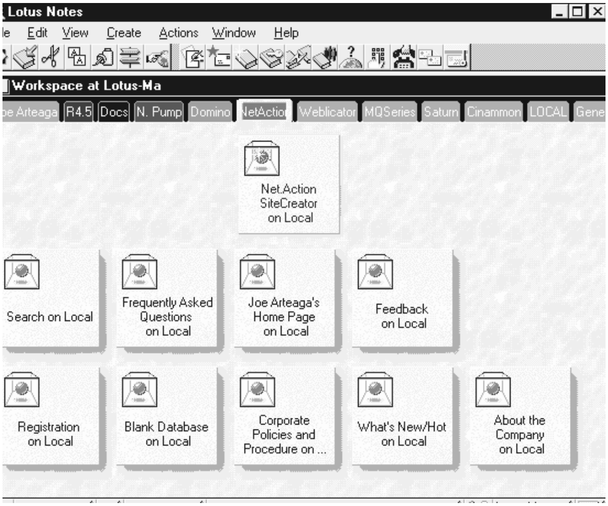
To accomplish this, Domino.Action will bundle additional software and tools, including:

- SiteCreator, for selecting and configuring the key elements of the site: home page; about-the-company; press release library; job postings; customer feedback; policies and procedures manuals; discussions; visitor registration.
- Editorial production tools: RTF- and ASCII-to-Notes / HTML translation, a version of the tools used by publishers on Notes:Newsstand.
- Command and control for the 'net. The Domino environment for application development provides a firm but flexible structure for electronic publishing that solves an enormous problem of creating content on the 'net: At last, the design of an electronic publication on a Web site can be separated from the creation and management of its contents. Domino.Action supports the design of content applications that can be displayed in both the Notes client and a Web browser. Documents are saved to the application, and displayed in the proper form when retrieved. This means:
    - Content-creation privileges can be set within the application, and authorship of the application's content can be distributed via the 'net. Domino administers security and controls access, and Notes replication can be used at any point in the process, from collecting raw text and images for editing to placing finished documents on the server.
    - Design standards can be established and enforced. Consistency is an automatic outgrowth of the design process. No longer is the look and feel of the page or the site at the mercy of the last person who edited the HTML code. The authors and editors no longer need to be experts at CGI scripting or UNIX administration. Any user who can create e-mail can create content for a 'net site.

  - Management of the content can be as simple or as sophisticated as the organization requires. Workflow tools are available for the creation of editing and approval processes, audit trails and sign-offs. And the management inherent in Notes provides the ideal growth path for a Web site making it easier to manage today's complex multi-page sites.
- Interactive applications such as the discussion databases for two-way communication and dynamic content.

SiteCreator

Here is a look at the Web site databases created by the SiteCreator program:



Net.Marketing

Companies that want to do marketing and sales on the Internet will use the Net.Marketing package to add these functions to the Domino Server and Domino.Action package:

- SiteCreator for creating catalog; request for information; survey / questionnaire; lead capture, collateral library (including page templates for brochure, white paper, press release, spec sheet, price list); event scheduler / signup.
- Production / workflow tool.
- Marketing tools
  - mailing list manager / lead manager
  - catalog builder
  - order processing / shopping basket manager
  - payment processing

Net.Marketing provides templates for forms and questionnaires that let an organization gather information about that audience and its stated interests and deposit it in Notes or relational databases for aggregation and analysis.

Net.Marketing has functionality in the following areas:

- Reduces the time to market delivery of your customer brochures by managing the electronic publishing of material.
- Forms-based surveys also make qualification of customers an integrated step of the process.
- Inquiries can be routed to the proper salespeople and electronic dialogs initiated.
- Statistics can be generated to track interest in particular product lines and feedback can be collected.
- Completion of screening questionnaires can be used as triggers for granting access to sites and discussions.
- Sophisticated targeted marketing and advertising applications based on content usage analysis, whether third-party applications or developed in Notes.

Notes databases provide the foundation for creating and managing your virtual marketplace. Net.Marketing adds tools and templates for the easy creation of widely used applications, such as document libraries, catalogs, and electronic brochures. The catalog can be managed as a distributed application with controlled access and integrated approval workflow. Documents or other electronic products can be published or updated quickly. Controlling the user’s access and membership in groups through the server’s name-and-address book means that factors such as prices and discount percentages can be treated as variable data.

For processing the order, Net.Marketing supports a “shopping cart” metaphor that lets a user browse the available items, accumulating selections for purchase, then place an order for the items and, if appropriate, manage the delivery or route the order through an approval process. Net.Marketing includes all the elements required to accept payment information from the customer and validate the transaction with a clearing entity in a secure environment, and in real time. This means a user can enter a credit card number to “buy a subscription” to an information product and have immediate access to the product. In its first version Net.Marketing will provide access to a payment processing switch. When the Secure Electronic Transaction (SET) protocols sponsored by MasterCard, VISA, IBM, and Netscape, among other companies, are finalized, they will be supported as well.

The Scalability of Net.Marketing

Net.Marketing is aimed at the needs of organizations that want payment capabilities to close the loop of their online marketing activities, such as selling and delivering research reports and technical documents online, or accepting payment for a conference registration. But there are no inherent limits on the Net.Marketing tools, and a growth path exists for any catalog built with Net.Marketing from the integrated Notes / HTTP server to other IBM solutions, such as Net.Commerce.

Indigo

Indigo is an information delivery system based on Lotus' Domino technology, the Lotus Notes application engine and the Net broadcast technologies developed by PointCast Incorporated. Indigo is an authoring, communication and information delivery platform for corporate intranets that combines external news with internal communications.

Net.Service

Net.Service builds on the Domino Server and Domino.Action with:

- SiteCreator for creating: problem reporting; problem routing; knowledge base with query capability; catalog; documentation library (product updates, FAQs / tips and techniques); digital download.
- Link toolkit (to give users with Web browsers views into transaction / order processing systems).
- Service tools: reporting and analysis tools (to generate stats on visitor registrations / unique IDs, how many logged messages, for what products, how long to close, etc.)

One of the most critical applications for many organizations is customer service and support. Many companies have devoted extensive resources to building service applications on the workflow and knowledge base capabilities of Notes, and Net.Service allows them to bring their customers directly into the system via a Web connection. Net.Service supports applications that can collect information in forms and questionnaires, then act on it, directing the users to information sources, and routing problem reports to customer-service providers within the organization.

For organizations that don't yet use Notes, doing customer service on the Web may be the most important application to develop. A sizable installed base of applications exists that use Notes workflow for call tracking, and its document management to control knowledge bases. And a vast amount of expertise exists among the Notes Business Partners who created these applications.

As Internet phone software moves telephony onto the Web, Net.Service will be there to provide the management tools to integrate this new technology into a high-quality customer-care system. With these tools, Net.Service can be used to create a managed escalation path that guides the customer toward an answer, and creates a record of the process. The path can begin with databases of FAQs and tech notes, move through product updates and timely information, and, if necessary, generate e-mail requests that bring the customer into direct contact with an employee, with built-in status reporting all along the way until the problem is resolved.

The Strategic Direction of Lotus' Internet Applications

Lotus will continue to build on the Domino platform, using the integrated server model to enhance Notes' ability to address the broadest possible base of clients, and to give those clients maximum access to the server — including future versions of the Notes client. Other Internet Applications will build on providing access from a Web browser to the Domino server's application-creation functions, which will let ISPs offer point-and-click Web site creation to their customers, as just one example.

Lotus will continue to build upon Domino and Lotus' Internet Applications to bring the best interactive Web applications development and hosting server on the market.

Web Site Organization

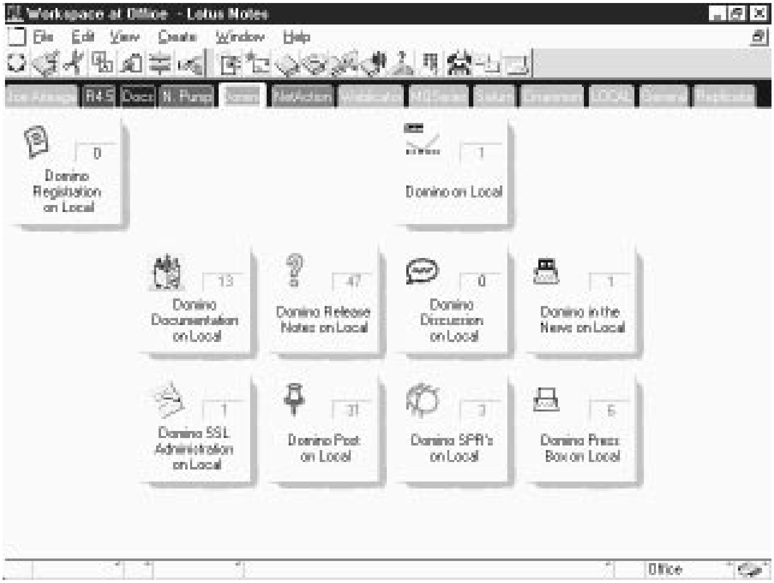
Your Notes application is usually accessed by opening the database from the server's directory through the Lotus Notes Workspace. With Domino, it can now be accessed by Web users as well as Notes clients through an Internet connection.

A Web site contains a home page and related links to help users move around the information and applications on your Web site. The home page is an HTML coded file with text and graphics that are linked to the other areas you want users to access.

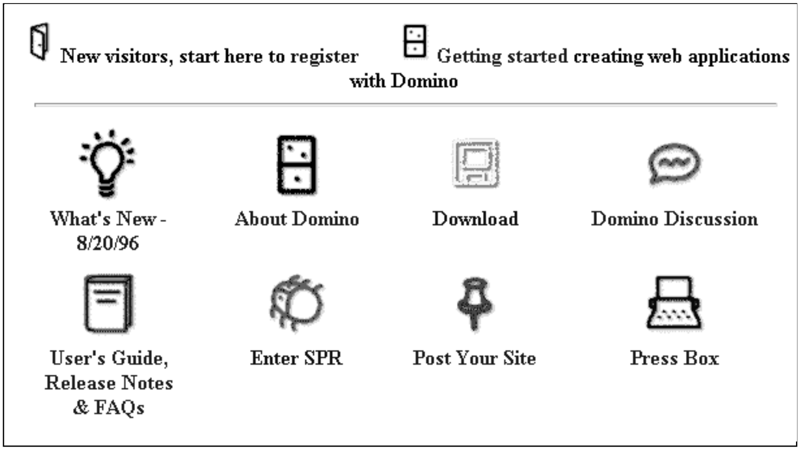
On traditional Web servers, the different pages and the associated compound elements are organized in hierarchical directory structures which apply the file system security available on that operating system platform. When you issue an HTTP request to see a page, you are opening a new HTML coded file existing in a directory.

With Domino, your Web site is structured through Notes databases designed in the Notes Object store format. When you issue an HTTP request to see a page, you are opening a Notes element through a Universal Resource Locator (URL) command and Domino is translating it for viewing as a Web page.

Therefore in Notes, your Web site is composed of one or more databases and structured Notes elements like views, documents, navigators and also embedded or linked Web standards like CGI, Java. For example, in the Notes workspace the Domino site looks like this:


Workspace at Office - Lotus Notes
File
Edit
View
Create
Window
Help
R4.5
Docs
N. Pump
NetAction
0
Domino Registration on Local
1
Domino on Local
13
Domino Documentation on Local
47
Domino Release Notes on Local
0
Domino Discussion on Local
1
Domino in the News on Local
1
Domino SSL Administration on Local
31
Domino Post on Local
3
Domino SPR's on Local
6
Domino Press Box on Local
Office

When accessing this "application" over the Web, you start off at a home page. The Domino Web site home page is displayed below. This page is actually the About Document of the Domino database which we saw on our workspace in the previous figure. It contains links to the other databases on our workspace.



New visitors, start here to register

Getting started creating web applications with Domino

What's New - 8/20/96

About Domino

Download

Domino Discussion

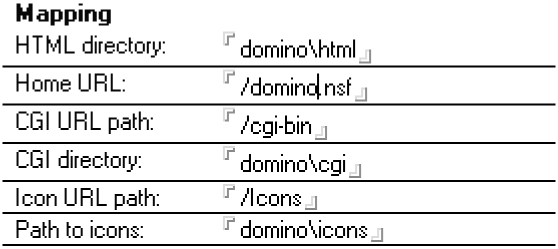
User's Guide, Release Notes & FAQs

Enter SPR

Post Your Site

Press Box

Since this will be the default page for all users to go to, you should also set the page to display automatically even when the user accesses the server directly by its site or host name. To do this you must set the ‘Home URL’ setting in the server document’s HTTP Settings section as shown below:



Mapping

HTML directory:	domino\html
Home URL:	/domino.nsf
CGI URL path:	/cgi-bin
CGI directory:	domino\cgi
Icon URL path:	/Icons
Path to icons:	domino\icons

In the next section we show you how to define a home page to display on your Web site using the Domino server.

Designing Your Home Page

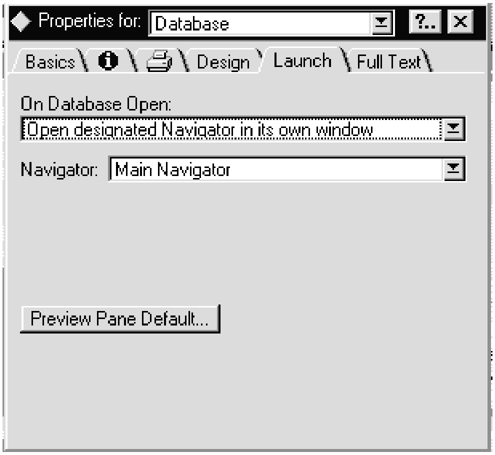
A home page is a Web site element which guides Web users through your site’s information and application(s). It also serves as the site’s focal point.

To create your home page:

1. Create your Site database.

This will contain your home page and links to information and applications in your Web site. This can be any database of your choice.

2. Set the database properties to launch your choice automatically when the database is accessed. For example to launch a navigator:



Properties for: Database

On Database Open: Open designated Navigator in its own window

Navigator: Main Navigator

Keep in mind that when setting On Database Open Launch in the Database Properties InfoBox, the Launch 1st attachment in the “About database” option is not supported.

Both navigator options, Open designated Navigator and Open designated Navigator in its own window, open a navigator in a separate window. The option Restore as last viewed by user is interpreted as View - Show Folders.

You may also specify an HTML file to serve as the home page.

Web Application Design Elements

The Domino server converts the Notes design elements to HTML to be displayed to the Web browser. This section deals with using both Notes elements and Internet technologies to create your Web application. If there are restrictions and differences from the normal Notes processing of Notes Elements, these are listed for your reference.

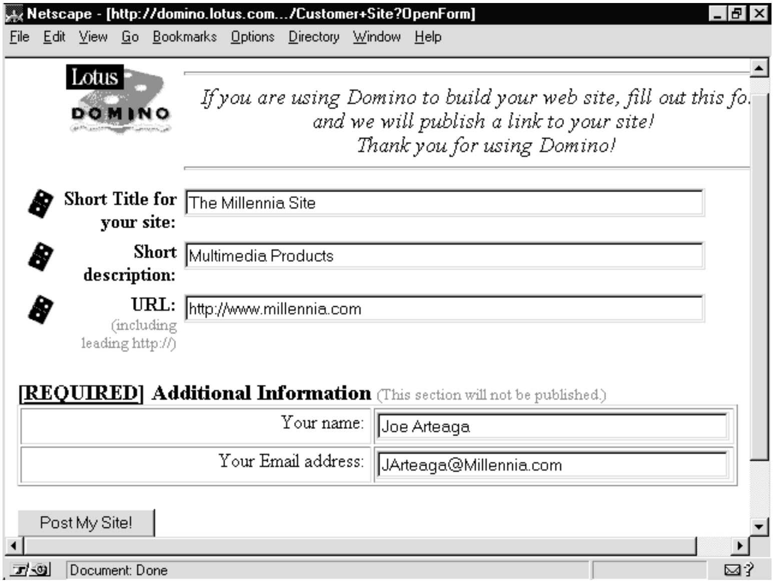
Web Forms

Domino lets you capture information that Web users enter in forms and store in Notes databases. When you create input forms, you can use Notes application development resources — for example, computed fields and input validation formulas. In addition, you can add HTML to further control the appearance of the form, capture the values in CGI variables, and customize the Web page response that users see when they submit a form.

To allow users to save documents, Domino places a Submit button at the end of each form when it converts the form to HTML.

Tip To create a customized Submit button in either a different location on the form or with a different label, choose Create - Hotspot - Button where you want to place the button and write the button label. Domino ignores all button formulas and treats all buttons as Submit buttons. If you create multiple Submit buttons on a form, Domino uses only the first button you created, and **only** that button appears on the Web.

For example, the following form from the Lotus Domino site allows Lotus to track Domino Beta Code. Notice the customized Submit button:


Netscape - [http://domino.lotus.com.../Customer+Site?OpenForm]
File Edit View Go Bookmarks Options Directory Window Help
Lotus
DOMINO
Document: Done

If you are using Domino to build your web site, fill out this fo
and we will publish a link to your site!
Thank you for using Domino!

Short Title for your site: The Millennia Site

Short description: Multimedia Products

URL: (including leading http://) http://www.millennia.com

[REQUIRED] Additional Information (This section will not be published.)

Your name:	Joe Arteaga
Your Email address:	JArteaga@Millennia.com

Post My Site!

Using $$Return to Create Customized Responses and Run CGI Scripts

After users submit a form, Domino responds with the default confirmation “Form processed.” To override the default response, add a computed $$Return field to your form and include HTML code as part of the formula for the field.

You can also use a $$Return field to run a custom CGI (Common Gateway Interface) program after the user submits the form and Notes creates the document. For example, you may run a CGI program that runs a NotesPump Activity document. The Web client displays the output of the CGI program to the user.

To run a CGI program, include the URL to the CGI program file and enclose it in brackets. Note that you can pass arguments — for example, values from fields in the form — to the CGI program. For example,

```
"[http://www.lotus.com/cgi-bin/register.exe?" + Email + "&&&"
+ LastName + "&&&]"
```

Note If input data from submitted forms can be processed in “batch” mode and messages to the user are not necessary, consider writing a LotusScript agent to further process the data instead of using a $$Return field.

Personalizing Responses

Create a personalized message for the user who submits a form. For example, the following $$Return formula returns the response “Thank you,” and appends the user’s name.

```
who:= @If(@Left(From; " ") = ""; From; @Left(From; " "));
@Return("<h2>Thank you, " + who + "</h2><br><h4>
<a href=/register.nsf/Main+View?OpenView>Main View</a>");
```

Adding a Link to Another Page

Include HTML with a URL in a response to link to another page based on field values in the submitted form. The following $$Return formula returns a response based on the region the user selects. For example, if the user selects Europe, the message “Visit our site in Italy” with a link to the Web site in Italy is shown.

```
@If(Region="Asia"; stdAnswer + "<h2>Visit our site in <a
href=\"http://www.japan.lotus.com\">Japan</a></h2>" +
stdFooter;

Region="Europe"; stdAnswer + "<h2>Visit our site in <a
href=\"http://www.lotus.com\it_ciao/it_ciao.htm\">Italy</a></
h2>" + stdFooter;
stdAnswer + stdFooter);
```

Returning Another Page

To jump to a different Web page after the user submits the form, enclose a URL for the page in brackets. When the user submits the form, the Web client displays the referenced document. For example, the following $$Return formula displays the home page for the Lotus Japan site.

```
"[http://www.japan.lotus.com]"
```

Using CGI Variables to Capture User Information Automatically

Common Gateway Interface (CGI) programs are a standard for interfacing external applications with HTTP servers. When a Web user submits a form, the HTTP server gathers the information and passes it to a CGI program. Although Domino doesn’t require the use of many types of CGI programs that a typical HTTP server requires, Notes application developers may want to capture CGI environment variables that give information about the Web client.

To return CGI variables, create an editable field with the corresponding variable name (as listed in the following table). Mark the fields “Hide when Editing,” so users cannot enter information in them. For example, to learn the Internet Protocol (IP) address of the user submitting the form, add a field named Remote_Addr to the form.

Field Name	*Returns*
Remote_Host	The hostname making the request.
Remote_Addr	The IP address of the remote host making the request.
Remote_User	Authentication method that returns the username they have authenticated as.
HTTP_User_Agent	The browser the client is using to send the request.
HTTP_Referer	The URL of the page the user used to get here.
Server_Software	The name and version of the information server software running the CGI program.
Server_Name	The server's hostname, DNS alias, or IP address as it would appear in self-referencing URLs.
Server_URL	URL passed to the server.
Gateway_Interface	The version of the CGI spec to which the server complies.
Server_Protocol	The name and revision of the information protocol this request came in with.
Server_Port	The port to which the request was sent.
Request_Method	The method with which the request was made. For HTTP, this is "GET," "HEAD," "POST," etc.
Path_Info	The extra path information, as given by the client. In other words, scripts can be accessed by their virtual pathname, followed by extra information at the end of this path. The extra information is sent as PATH_INFO.
Path_Translated	The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.
Script_Name	A virtual path to the script being executed, used for self-referencing URLs.
Query_String	The information which follows the ? in the URL which referenced this script.
Auth_Type	If the server supports user authentication, and the script is protected, this is the protocol-specific authentication method used to validate the user.
Remote_Ident	This variable will be set to the remote user name retrieved from the server. Usage of this variable should be limited to logging only.
Content_Type	For queries which have attached information, such as HTTP POST and PUT, this is the content type of the data.
Content_Length	The length of the content as given by the client.
HTTP_Accept	The MIME types which the client will accept, as given by HTTP headers.

When putting together a form, keep the following information in mind.

Form Properties

Some form properties behave differently over the Web:

- Merge replication conflicts setting is not supported.
- Version control is not supported.
- Automatically refresh fields setting is not supported.
- Disable field exchange is not applicable to Web users.
- On Create event
  - Formulas inherit values from selected document is supported, except for rich text field inheritance on forms.
  - Because document selection isn't applicable to the Web, default value formulas cannot reference a "selected document in the view."
  - Inherit entire document into selected rich text field is not supported.
- On Open event
  - Show context pane is not supported.
- Store form in document is supported in read mode only. Do not use for documents that need to be created or edited on the Web.
- Security
  - Default encryption keys are not applicable to Web users.
- Disable printing/forwarding/copying from clipboard is not supported.

Form Elements

Domino 1.0 has some limitations we discuss here, but this is quickly changing. In future releases these will be resolved. Check the Lotus Domino Web site for the latest release.

Attachments are supported over Web forms, including attachments with no hotspots.

On forms, buttons are not supported, except for customized Submit buttons. See the Section on customizing the submit button in this chapter.

Graphics are supported and displayed on the HTML page.

Hotspots are supported, except for pop-ups. Layout regions are not supported. You can use tables to align form components.

Sections are supported, but they always appear expanded and without a title. Sections cannot be hidden and section security features, such as editor properties and signing, are not available.

Tables are supported, including setting the width of table columns. The Notes table width and column width information is used on the Web. If the top left cell of a Notes table has a border, the entire table is displayed with a border; otherwise, there is no border.

Fields on Web Forms

Keyword fields are supported, except for the keyword entry helper option. You can control the number of visible rows by adding HTML code to the field help in the Field Properties InfoBox. See the section on adding HTML attributes in this chapter.

Formatting Text for Web Applications

Domino supports most Notes formatting features by converting them to HTML tags. Certain Notes formatting features (for example, indentation, inter-line spacing, and tabs) do not appear when viewed from a Web browser because HTML has no corresponding format. In addition, not all browsers support all the HTML tags that Domino generates.

When you format text remember that Domino maps the font size you select in Notes to HTML sizes. Point sizes equal to or less than 8,10,12,14,18,24,>24 map to HTML sizes 1,2,3,4,5,6, 7 respectively.

Note Domino does not map font sizes to HTML headings.

Choosing Font Style

Domino supports all Notes font styles. To align a column of numbers or preserve or insert spaces, use a monospaced font like Courier. Domino converts monospaced fonts to a monospaced font on the Web and preserves any spaces you enter.

Choosing a Text Color

Domino supports all text colors available in Notes.

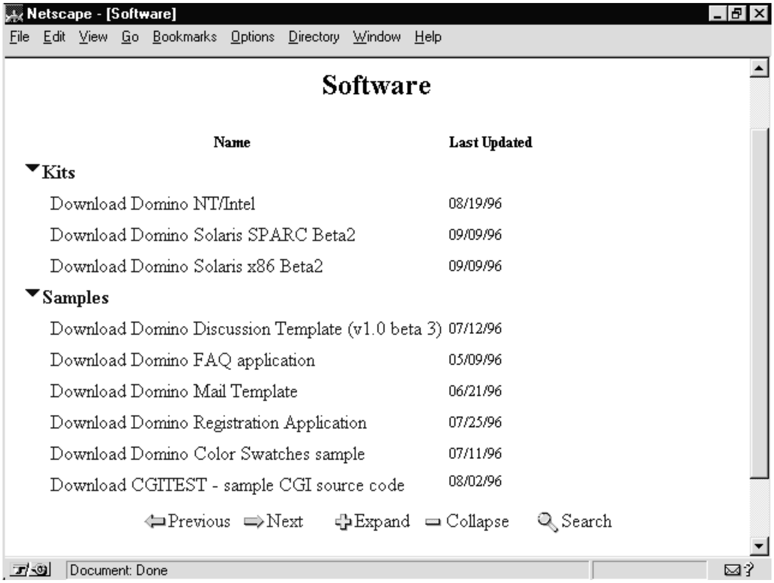
Choosing Other Formatting Features

Domino supports the following:

- Bullets and numbered lists
- Alignment (except full justification and no wrap)
- Spacing
- Named styles

Web Views

Notes Views are dynamically created as Web pages to display the links to the documents. You can move around the view, expand and collapse, and search much as you do in Notes except for the fact that the view is split into pages. This is so that if a view contains hundreds of documents, you are not presented with one page containing all of them. This gives you better performance on the page you are looking at and it makes navigation more manageable. A sample view is shown below:


Netscape - [Software]
File Edit View Go Bookmarks Options Directory Window Help
Previous Next Expand Collapse Search
Document: Done

Software

Name	Last Updated
▼Kits	
Download Domino NT/Intel	08/19/96
Download Domino Solaris SPARC Beta2	09/09/96
Download Domino Solaris x86 Beta2	09/09/96
▼Samples	
Download Domino Discussion Template (v1.0 beta 3)	07/12/96
Download Domino FAQ application	05/09/96
Download Domino Mail Template	06/21/96
Download Domino Registration Application	07/25/96
Download Domino Color Swatches sample	07/11/96
Download CGITEST - sample CGI source code	08/02/96

The View shown above lists the documents on the Domino Download database. Notice the Expand and Collapse navigation buttons allowing you to use the view as you would in Notes.

Some key design points for views follow:

View and Folder Properties

- Folders are supported but you cannot move documents into folders.
- Multi-line column headings and multi-line rows are supported.

  Tip To prevent line wrap, specify 1 in the Lines per heading setting in the View properties InfoBox. (They'll have a <NOWRAP> HTML tag.) Specifying a number greater than 1 causes lines to wrap on the Web. The same guidelines are true for the Lines per row setting.
- On Open, Go To options are not supported.
- On Refresh options are not supported.

- Show in View menu is not applicable to Web users since Web applications do not have a View menu. To remove a view from the folders navigator, use a hidden view.
- Style options for Unread rows, Alternate row colors, Show selection margin, and Beveled column headings are not supported.
- View Indexing options are not applicable, views can be re-indexed at a Notes server.

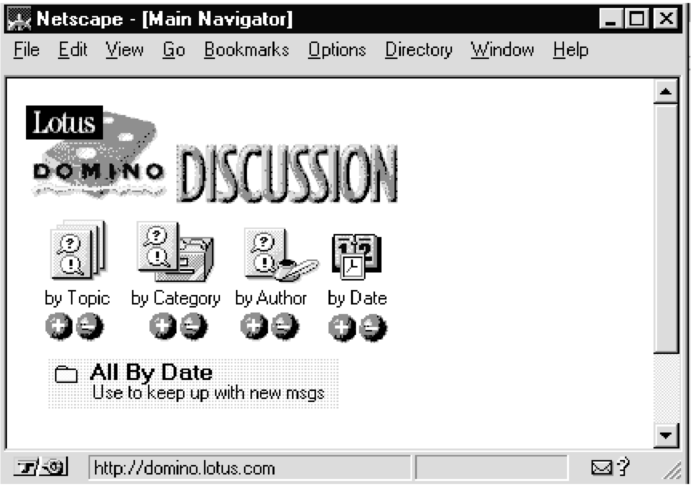
Column Properties

- Click on column header to sort setting is not supported.
- Collapsed / expanded categories setting is supported. It expands or collapses processes one category at a time (equivalent to Expand / Collapse Selected Level).
- Resizable columns are not supported.
- “Show twisties” when row is expandable is not available.

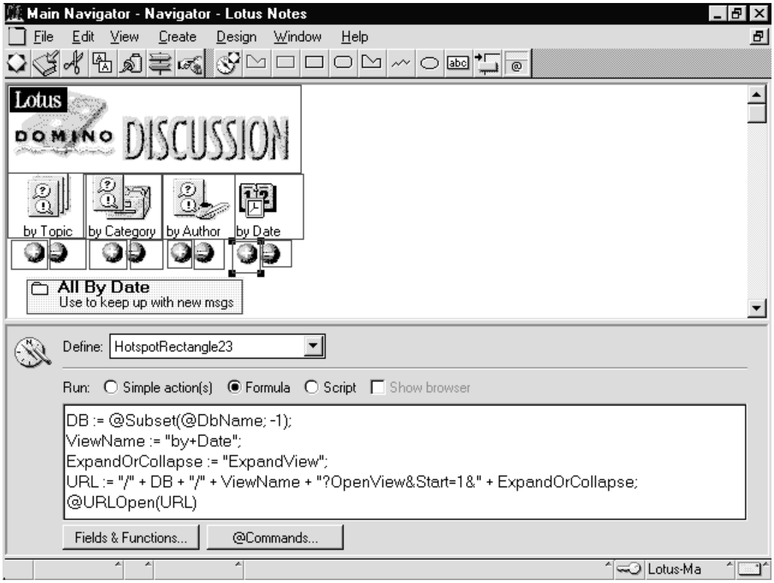
Navigators

Navigators are translated to image maps by the Domino server. Image maps are graphical areas on a Web page that contain links to other pages. When you open a navigator the different regions of the map correspond to a different Web page as defined in the navigator links. The navigator must be made up with a single graphic background with hotspots. Overlaid text and buttons are not supported. If you create a navigator that contains objects not supported on the Web, such as buttons, you see a bitmap, but the navigator is not functional.

The following is the navigator used in the Domino discussion database:



Here is the actual design showing the formula for one of the hotspots which expand a view:



Define: HotspotRectangle23

Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

```
DB := @Subset(@DbName; -1);
ViewName := "by+Date";
ExpandOrCollapse := "ExpandView";
URL := "/" + DB + "/" + ViewName + "?OpenView&Start=1&" + ExpandOrCollapse;
@URLOpen(URL)
```

Client-Side and Server-Side Image Maps

Web image maps can be handled in two ways. With server-side image maps, the browser sends the coordinates for a region in the image map to the server, and the server sends the corresponding URL information back to the browser. With client-side image maps, the browser generates the URL information for the regions in the image map.

Domino converts a navigator into an image map that is both a client-side and a server-side image map, which means that all browsers are automatically supported. If the browser supports client-side image maps, it uses those tags. If the browser does not support client-side image maps, it uses the server-side image map HTML tags.

Use formulas to compute which view or database to link to. When you create the navigator, use actions for different regions. For example, use a formula to determine whether the user is a Notes user or a Web user. Then based on the result of the formula, use the @Command(OpenView) function to link to a specific view.

The following are property differences when using navigators over the Web:

- Auto adjust panes at runtime do not work as they are not applicable to Web users.
- Highlight when options for navigator objects is not supported.
- Navigator graphic background is supported over the Web. Choose Create - Graphic Background.

  Note Do not use the Create - Graphic Button as it does not convert the navigator to an image map.
- Navigator objects are supported only as hotspot polygons and hotspot rectangles. All other objects, such as text boxes, ellipses, and graphic buttons, are not supported and do not function.

Agents, LotusScript, and Actions

The following list describes the restrictions placed on using actions, agents and LotusScript with your Web applications.

- Agents are supported, except for the run option "If Document Has Been Pasted" and the document selection option "Selected documents." The concepts of "pasted documents" and "selected documents" don't apply to Web applications.
- LotusScript for forms, actions, and buttons is not supported. Use the $$Return field and CGI scripts to achieve the same results as a LotusScript QuerySave or QueryClose event.
- Simple actions for agents, form and view actions are not supported.
- System actions supplied with forms and views (such as @Edit Document, @Categorize) are not supported. Use supported @Commands to create the equivalent actions.

@Function Formulas in Web Applications

Using @Functions in Web applications is somewhat restricted. The following list describes those @Functions not available or where available, the restriction on using it over the Web is noted.

- Hierarchy Functions
  - @AllChildren, @AllDescendants, @Certificate, @DBCommand, @DocChildren, @DocDescendants, @DocLevel, @DocNumber, @DocParentNumber, @IsCategory, @IsExpandable, @Responses,
  - @DocSiblings is available for use only in view and column formulas.
- Dynamic Data Exchange Functions
  - @DDEExecute, @DDEInitiate, @DDEPoke, @DDETerminate.

- @Environment, @SetEnvironment, ENVIRONMENT keyword not applicable to Web users. Use predefined field names to gather information about the Web user's environment by requesting Common Gateway Interface (CGI) environment variables.
- Mail Functions
  - @MailSend, @Domain, @MailDbName
- Preferences Functions
  - @MailEncryptSavedPreference, @MailEncryptSendPreference, @MailSavePreference, @MailSignPreference, @GetPortsList
- Element state Functions
  - @IsAgentEnabled
- @UniqueID is supported. Avoid values based on time computations in computed-when-composed fields, such as @Now and @UniqueID, that may be updated a second time during a Web transaction. To simulate an @UniqueID formula, use @DocUniqueID and compute an extra value, such as an incremental integer.
- Web Navigator Functions
  - @URLGetHeader, @URLHistory , @UserAccess
- Security Functions
  - @UserPrivileges
  - @UserRoles is supported. Appends $$WebClient to the list of roles and can be used to differentiate between Notes and Web clients
- @ViewTitle is supported on forms in read and edit mode. Newly composed forms cannot use it.
- @Platform is supported and returns server's platform only.
- @DocMark
- @DeleteDocument
- @DialogBox
- @PickList
- @Prompt
- @IsModalHelp

@Commands Formulas

Most @Commands are based on the Notes workstation user interface and are not applicable to Web applications. The following @Commands are supported with the noted differences. They are converted to URLs on the Web:

@Function Name	*Description*
@Command([Compose])	Can be used but the server argument cannot be used.
@Command([EditClear])	Deletes the current document and can be used only for forms.
@Command([EditDocument])	Can be used only on forms.
@Command([FileOpenDatabase])	Accepts only the database, view, and navigator name arguments.
@Command([NavigateNext]), @Command([NavigatePrev]), @Command([NavigateNextMain]), @Command([NavigatePrevMain])	Navigation Commands
@Command([OpenNavigator])	Accepts only the navigator argument.
@Command([OpenView])	Accepts only the view name argument.
@Command([ToolsRunMacro])	

Working With Images

Domino converts images to Graphics Interchange Format (GIF) or Joint Photographic Experts Group (JPEG) files for display on the Web based on server document setting. Domino passes the image size and scaling information to the browser. If the browser supports scaling, the image has the same size and scale as it does in Notes. If the browser does not support scaling, the image appears in its original size, regardless of how you size it in Notes.

Notes stores graphics in two formats — a platform-dependent metafile and a 256-color platform-independent bitmap. Domino uses the platform-independent bitmap which, in some cases, may cause the graphic to look slightly different on the Web.

Interlaced GIF Files

Browsers typically display a GIF image while it is being loaded. An interlaced GIF file is one whose image lines are stored out of sequence — for example, as every eighth row, then every fourth row, then every second row, and so on. To users, the image seems to appear quickly because their eyes tend to “fill in” the missing pieces.

Progressive JPEG Files

Browsers typically load and display a JPEG image in one pass. A progressive JPEG image loads incrementally in several passes: the image becomes clearer with each pass. The effect is that users can identify a progressive JPEG image before it is completely downloaded.

Using Passthru HTML to Reference an Image

You can use passthru HTML to reference an image for Web-only viewing. For example, to reference a GIF file from the IBM home page, use this formula:

```
[<IMG SRC="http://www.ibm.com/dom.gif"WIDTH=80HEIGHT=80>]
```

The current Domino URL syntax for referencing bitmaps in Notes documents — specifically, the field offset part — makes it impractical for users to create these URLs manually. As an alternative, you may paste the actual bitmap in place of a reference or create URL references to image files stored in the file system.

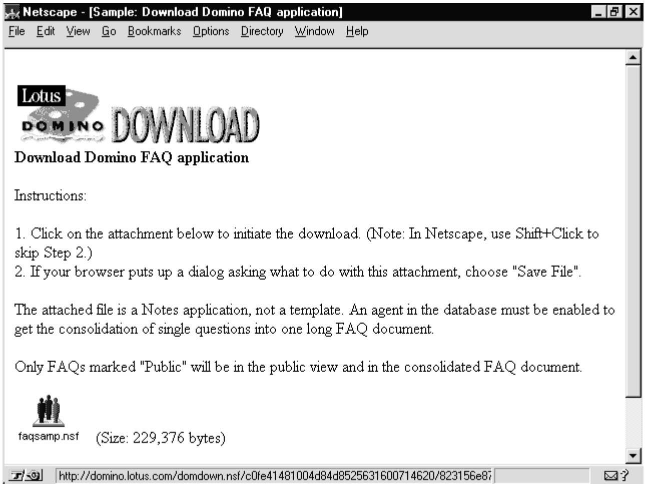
Working With Attachments

You can attach any type of file to a document, including binary files, compressed files, executable files, and Notes database (NSF) files. The reference to the file attachment appears as a graphic of the file and is located in a rich text field in Notes.

Domino converts the graphic representing the file to a GIF image and displays the GIF image as a link to the attached file on the Web. When Domino converts a file attachment, it generates a URL for the file attachment. The last component of the URL is the file name. For example:

```
http://domino.lotus.com/domdown.nsf/<ViewUNID>/<DocumentUNID>
/Attachments/0.54/Dom.txt?OpenElement
```

Here is a sample document containing an attachment. Notice the URL on the bottom of the screen:


Netscape - [Sample: Download Domino FAQ application]
File Edit View Go Bookmarks Options Directory Window Help
Lotus
DOMINO DOWNLOAD

Download Domino FAQ application

Instructions:

1. Click on the attachment below to initiate the download. (Note: In Netscape, use Shift+Click to skip Step 2.)
2. If your browser puts up a dialog asking what to do with this attachment, choose "Save File".

The attached file is a Notes application, not a template. An agent in the database must be enabled to get the consolidation of single questions into one long FAQ document.

Only FAQs marked "Public" will be in the public view and in the consolidated FAQ document.

faqsamp.nsf (Size: 229,376 bytes)

http://domino.lotus.com/domdown.nsf/c0fe41481004d84d8525631600714620/823156e87

The browser uses the name specified as part of the URL when it saves the file.

Launching the Application for an Attached File

Domino supports MIME (Multimedia Internet Mail Extension) type mappings — a process that maps an attachment's file extension to an external viewer or a helper application. This means that users can view or launch attachments automatically from Web pages.

For example, a .WAV file on a Web page can be played automatically on a user's machine. As long as the application for the attached file is installed on the user's machine and the user's HTTPD.CNF file includes a mapping for .WAV files, the browser starts the Media Player when a user clicks the attached file. Refer to the Domino Documentation for more on MIME support.

Adding HTML to Notes Elements

To apply formats and attributes that are available in HTML but not in Notes, include HTML code in Notes documents, forms, fields, or view columns. Domino does not attempt to convert the data; instead, it combines the HTML code with the Notes data and passes it through to the browser for display.

To enter HTML code, you can:

- Enter the HTML instructions directly in a document, a form, a field or elements in a view.
- Create and apply a paragraph style named HTML.

HTML Code Syntax

To enter HTML code, use the following syntax:

```
[<html code>]
```

where *html code* is the HTML tag you want to include.

Useful Examples

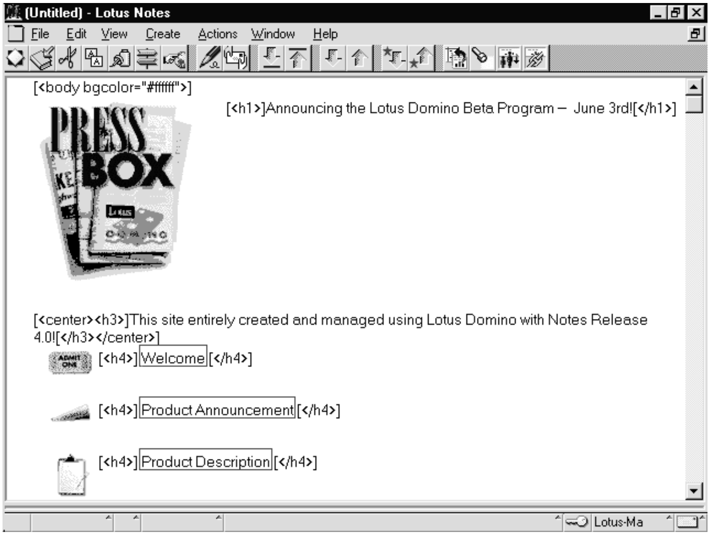
The following table lists some of the more useful HTML codes:

HTML Code	*Description.*
<h1> thru <h8>	Headings from 1, the largest to 8 the smallest. End a heading with the </h1> command.
<hr>	Creates a horizontal rule across the page.
<p>	Paragraph
<br>	Insert a break or new line
<a href>	Link to another page. <a href="http://www.ibm.com">IBM Home Page</a>

To add a horizontal rule to your page when viewed on the Internet, enter:

```
[<hr>]
```

Here is a sample document in Notes which contains HTML code:



[<body bgcolor="#ffffff">]

[<h1>]Announcing the Lotus Domino Beta Program – June 3rd![</h1>]

[<center><h3>]This site entirely created and managed using Lotus Domino with Notes Release 4.0![</h3></center>]

[<h4>]Welcome[</h4>]

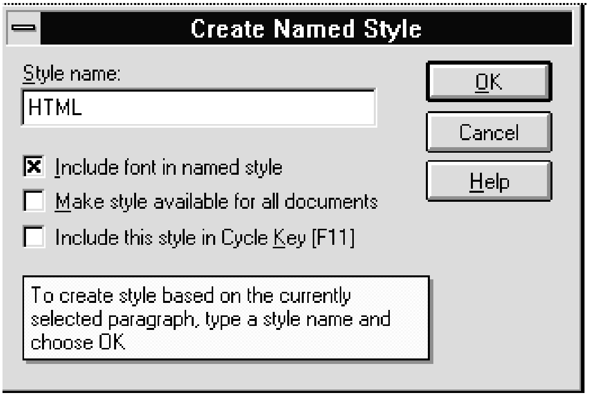
[<h4>]Product Announcement[</h4>]

[<h4>]Product Description[</h4>]

Creating an HTML Text Paragraph Style

To include HTML code in a Notes document or form, create a text paragraph style and name it HTML. Domino does not convert anything formatted with the HTML paragraph style.

1. Highlight the paragraph you want to format as HTML.
2. Choose Text Properties and click the Styles tab in the InfoBox.
3. Click Create Style, enter HTML in the Style Name field, and click OK.



Style name: HTML

☒ Include font in named style
☐ Make style available for all documents
☐ Include this style in Cycle Key [F11]

To create style based on the currently selected paragraph, type a style name and choose OK

4. Close the InfoBox.

Tip When you use the HTML paragraph style, it's not necessary to enclose the HTML instructions in brackets.

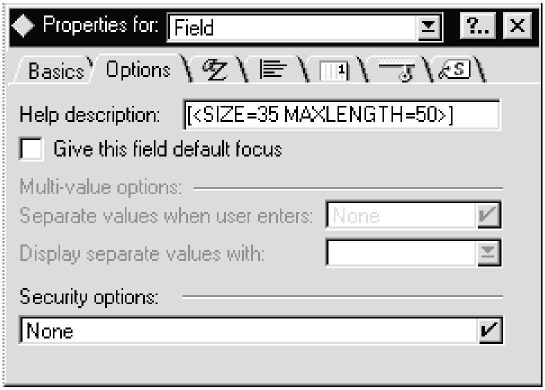
Tip To hide the HTML code from users who read the document, select the text, choose Text - Text Properties, select the Hide Paragraph tab, and select "Previewed for reading" and "Opened for Reading."

Adding HTML Attributes to an Editable Field

Domino lets you control the size and length of fields in the form. To override the field defaults, enter HTML code in the *Help description* box of the Field Properties InfoBox.

Size and Maximum Length of Text Fields

You can specify the size and length of text fields. For example, the following figure specifies that the Name field display 35 characters and accept a maximum of 50 characters.



Help description: [<SIZE=35 MAXLENGTH=50>]

□ Give this field default focus

Multi-value options:

Separate values when user enters: None

Display separate values with:

Security options:

None

Maximum Number of Visible Choices in Keywords Fields

You can specify the maximum number of rows that are visible in keywords fields. For example, the following command specifies that the Request field has 5 rows. If that field has more than five keywords, a scroll bar appears so users can scroll to other keywords.

```
[<SIZE=5>]
```

Wrap Setting for a Rich Text Field

You can specify the wrap setting for a rich text field. For example, the HTML tag [<WRAP=VIRTUAL>] specifies that text in the field wrap and that no line feeds or carriage returns be inserted at the end of lines.

Adding HTML Code to a View

You can include passthru HTML in a view column formula that displays an image depending on the results of the formula. For example, the following formula inserts a NEW.GIF image if the document was created within the last 5 days; otherwise, no graphic is displayed.

```
@If(@Now>@Adjust(@Created;0;0;5;0;0;0);"";"[<img
src=/gifs/new.gif no border >]")
```

Creating Links

Creating and maintaining links on Web sites is easy because you can use Notes links to generate structured, easy-to-navigate, accurate links that don't require manual hard coding or continual updating in the file system.

Domino converts any Notes-created links (document, view, and database links) to hypertext links that allow users to browse linked pages on the Web. In addition, you can include links to other pages on the Web.

Links in Notes are more stable than links in HTML as they are based on the page's identity not on its location in the file system. HTML links are hard coded references to file names, so if you move or rename a file, the link no longer works. Notes links do not reference file names; rather, they reference the universal ID (also known as the unique ID or UNID) of a database, view, or document. So even if a view name changes, a link to a document in that view remains valid.

Linking to Documents, Views, and Databases

A Notes Link

Add a Notes link to a document to let users switch to another document, view, folder, or database. In the Web browser, users click the link icon to access the link.

A Link Hotspot

Add a link hotspot to an area of a document (such as text or a graphic) to let users switch to another document, view, folder, or database.

An Action Hotspot

Use an action hotspot with an @Command formula or an @URLOpen formula to create the link. For example:

```
@URLOpen("http://www.lotus.com")
```

An Action Bar

Use an @Command formula in the action bar to create the link. For example:

For a view in the current database

@Command([OpenView]; "*viewname*")

For a view in another database

@Command([OpenView];"":"*database*"; "*viewname*")

Passthru HTML

Use passthru HTML to link to any Web page. For example:

To open the By Date view in the Domino Discussion database:

```
[<AHREF=http://domino.lotus.com/disc.nsf/By+Date?OpenView>SHo
wTheByDateView</A>]
```

Linking to Forms and Navigators

To create a link to a form or navigator, use any of the following techniques:

Action Bar

Use an @Command formula in the action bar to create the link.

For a form in the current database:

```
@Command([Compose]; "formname")
```

For a form in another database:

```
@Command([Compose];"":"database"; "formname")
```

For a navigator in the current database:

```
@Command([OpenNavigator]; "navigatorname")
```

For a navigator in another database:

```
@Command([FileOpenDatabase];"":"database"; "navigatorname")
```

The following action button formula links to a Main Topic form:

```
@Command([Compose];"Main Topic")
```

Action Hotspot

Use @Command formulas or the @URLOpen formula in the action hotspot to create the link. For example, @URLOpen(“http://www.ibm.com”).

Note that in general, action hotspots and action bars accept the same types of formulas.

HTML Link

You can also use an HTML link. For example, the syntax for linking to a form is:

```
[<A HREF="/dbname/formname?OpenForm">Click here to create
document</A>]
```

Note Domino cannot generate links based on the currently selected document in a view because at that point there is no notion of a “selected” document to which to respond. Users must open (or select) a document to see the link to the Response form.

Linking to an External Web Site

To create a link to an external Web page, use any of the following techniques:

Action Hotspot
Use an action hotspot to let users perform the same action in Notes and on the Web. Use the @URLOpen formula in the action hotspot to create the link.

Action Bar
Use the @URLOpen formula in the action button to create the link. For example:

```
@URLOpen("http://www.ibm.com")
```

Passthru HTML
Use passthru HTML to link to any Web page.

```
[<A HREF=http://www.lotus.com>Go to Lotus</A]
```

Domino URLs

Domino supports a wide range of commands that can be used by the application developer to access the Notes database currently being viewed from a Web browser. Each command takes the form of a URL and allows you to access specific items within the database directly.

All calls to a Domino server from a Web client are made using the URL syntax. This is broken down into a number of component parts, the number being dependent on the Notes item you wish to access. All URL calls to Domino must begin with the server’s name and the protocol to use.

```
http://Millennia.Lotus.Com
```

In this case, the URL specifies which protocol to use, *HTTP* and then the name of the server, *Millennia.Lotus.Com*.

After this, you can specify parameters to Domino that follow the following syntax:

```
TypeOfNotesObject/NameOfNotesObject/
?ActionToPerform&Arguments
```

- Here, TypeOfNotesObject can be a database, a view within a database, a form within a database and so on.
- NameOfNotesDataObject is the name of the database you are trying to open, the name of a view within a database and so on.

- ?ActionToPerform is the type of action that you want to perform on the NotesObject such as opening the database, opening the view, creating a new document and so on.
- &Arguments are optional parameters that you can add to the URL to provide view formatting and document control.

Domino Objects

Below are the most commonly used cases for each of the objects.

Database Object

By default, when a Web client accesses a Domino site using just the site name, Domino will open the database that is specified in the server document of the Public Name and Address Book.

For example, when a Web client specifies HTTP://Millennia.Lotus.Com, Domino will open the database specified in the *Default home page* field of the Name and Address Book.

If you wish to access a different database you can specify the Database option in the URL, for example:

`http://Millennia.Lotus.Com/Courses.nsf/?OpenDatabase`

View Object

To open a specific view within a database, specify its name in the URL, for example:

`http://Millennia.Lotus.Com/Courses.nsf/CourseInfo/?OpenView`

This will open the CourseInfo view in the Courses database.

You may also specify the NoteID, NoteUNID or $defaultView instead of its name.

Form Object

To open a specific form within a database, use the following syntax:

`http://Millenia.Lotus.Com/Courses.nsf/CourseForm/?OpenForm`

This will open the CourseForm form within the Courses database.

You may also specify the NoteID, NoteUNID or $defaultForm instead of its name.

Navigator Object

To open a specific navigator within a database, use the following syntax:

`http://Millennia.Lotus.Com/Courses.nsf/MainNav/?OpenNavigator`

This will open the MainNav navigator within the Courses database.

You may also specify the NoteID, NoteUNID or $defaultNav instead of its name.

Agent Object

To run an agent on the server, use the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/UpdateAll/?OpenAgent
```

This will run the agent UpdateAll within the Courses database on the server.

You may also specify NoteID or NoteUNID instead of the agent's name.

Document Object

Documents need to be opened from a specific view within the database with the NoteID or NoteUNID of the document. To open a document within a database, use the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/CourseInfo/5ce629c6114
98d5085256392006a03e1/?OpenDocument
```

This will open a specific document within the Courses database displayed from the CourseInfoView.

Database Icon Object

The database icon can be referenced using the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/$icon
```

Database Help About Object

The database Help About document can be referenced using the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/$about
```

Database Help Object

The database help document can be referenced using the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/$help
```

Search and Search Form Object

A database can either be searched using the default search form or a form that you specify. To access the default search form use the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/CourseInfo/$searchForm
?SearchView
```

This will open up the default search form and allow users to search the CourseInfo view.

To specify your own document that users can enter search criteria into, use the following syntax:

```
http://Millennia.Lotus.Com/Courses.nsf/CourseInfo/
CourseSearch/CourseSearch?OpenForm
```

This command will open a document that contains the search fields you wish the user to enter information into. When the user then presses the submit button to start the search use the following syntax to execute their query:

```
http://Millennia.Lotus.Com/Courses.nsf/CourseInfo/
CourseSearch/?SearchView&Query=Notes+Expert+AND+Basingstoke
```

This command tells Domino to search in the CourseSearch view for all the documents containing the words “Notes Expert” and “Basingstoke.”

Domino Actions

As you have seen in the examples above, including the object in the URL is not always enough for Domino to know what it is you wish to do with the object. This is handled using actions, which make something happen to the object.

In Domino there are three types of actions, None, Implicit, and Explicit.

Implicit actions assume to act on the object in the URL, explicit actions force a specific action to occur.

Implicit Actions	*Explicit Actions*
?Open	?OpenServer
	?OpenDatabase
	?OpenView
	?OpenNavigator
	?OpenForm
	?OpenAgent
	?OpenDocument
	?OpenElement (bitmaps, attachments etc.)
?Create (documents only)	?CreateDocument (form post action)
?Edit	?EditDocument
?Save (documents only)	?SaveDocument (form post action)
?Delete (documents only)	?DeleteDocument
?Search (views only)	?SearchView

Domino Arguments

A further level of control can be given to the object and action that you specify in the URL. Each argument needs to be prefaced with an & symbol.

Below is a table of the available Domino arguments.

Related Action and Argument	*Function*
?OpenView&Start	Opens a view with a specified number of documents from the top, for example, ?OpenView&Start=3
?OpenView&Count	The number of documents displayed in a view
?OpenView&Expand	Expands an item in the view
?OpenView&ExpandView	Expands all items in the view
?OpenView&Collapse	Collapse an item in the view
?OpenView&CollapseView	Collapses all items in the view
?OpenElement&FieldElemType	Specifies the type of object expected at a field offset
?OpenElement&FieldElemFormat	Specifies the format desired for inline graphics (GIF or JPEG)
&OldSearchQuery	Re-runs the last query
?SearchView&SearchMax	Specify the maximum number of results you wish returned from the query
?SearchView&SearchWV	Include word variants in the search
&SearchOrder	Ascending or Descending
&SearchThesarus	??
&ParentUNID	Specifies the UNID of the document to respond to and/or inherit from when creating new documents

Chapter 15
Domino: Sample Applications

Overview

This chapter describes in detail three examples for using Lotus Notes with the Internet.

- A look at a Domino site registration database
- A look at the PageMinder agent
- A case study that builds a Web site to publish and retrieve information to and from the Internet.

This chapter was written using Release 1.0 of the Domino server.

An Application to Register Users Over the Web

If you want to protect certain areas of your site, and require that Web users enter a user name and password before they access protected information, for example, you'll need to create an account for each user. To do this, you register users in the Public Name and Address Book on the Domino server.

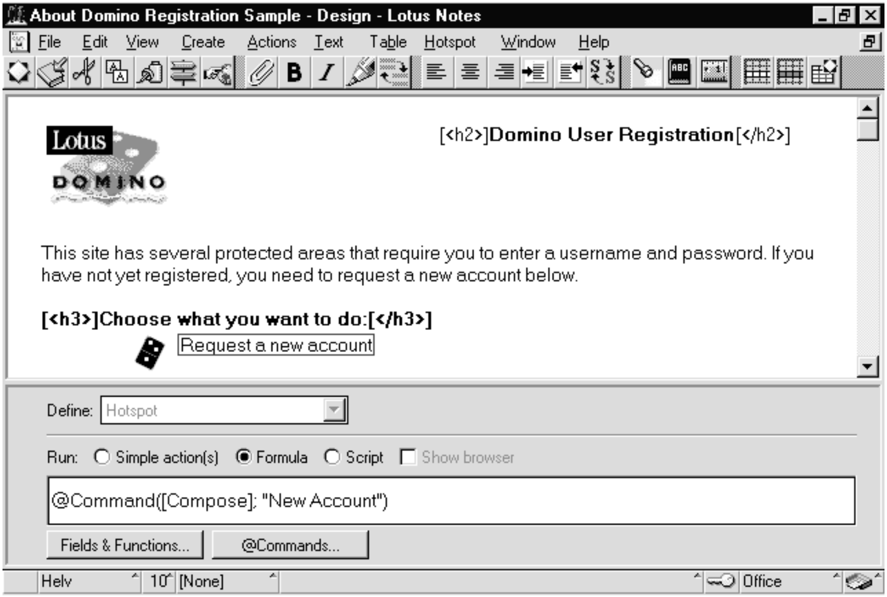
This section covers the design of such a Notes application. The following examples are taken from the *Domino Registration Sample* which is used to register persons participating in the Domino Beta Programs on the Web.

Basic Concepts

The intention of this application is to gather information about the user who wants to access your Web site and Web applications. The user fills out a form and submits the form. Once you have all the data you need to decide whether the user may use the server or not you will run some authorization tasks and provide the user with a user ID and password combination. An agent runs against the database looking for new forms and adds new users to the Name and Address Book. Then, when a user wants to access your server, the system checks if this user has already registered and allows the user access.

Application Design

The sample application consists of a single Notes database. By default, when this database is opened it displays its *About Database Document*. This document contains a hotspot which kicks off the registration of a user.


About Domino Registration Sample - Design - Lotus Notes
File Edit View Create Actions Text Table Hotspot Window Help
Lotus
DOMINO
Helv
10
[None]
Office

[<h2>]**Domino User Registration**[</h2>]

This site has several protected areas that require you to enter a username and password. If you have not yet registered, you need to request a new account below.

[<h3>]Choose what you want to do:[</h3>]

Request a new account

Define: Hotspot

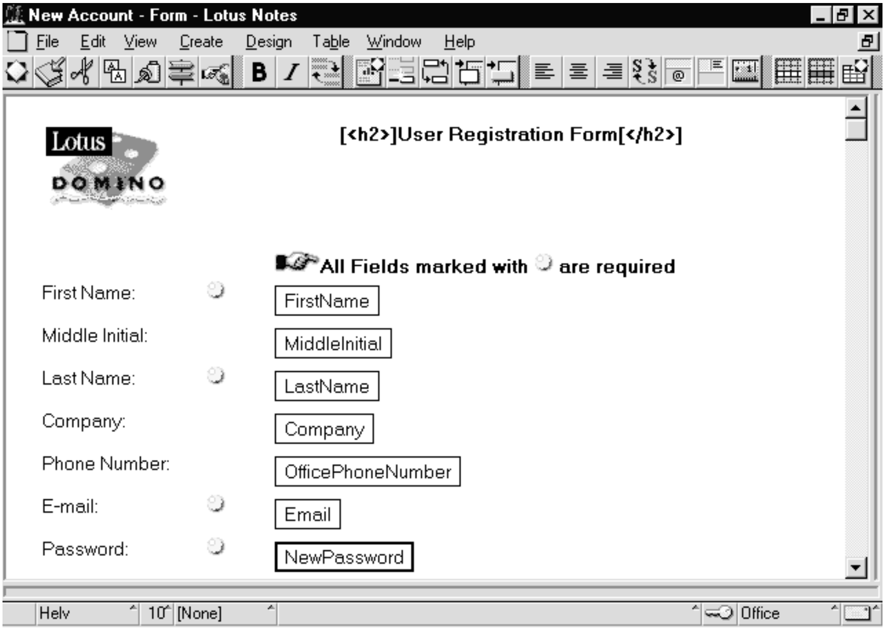
Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

```
@Command([Compose]; "New Account")
```

Fields & Functions...
@Commands...

Note The *About Database Document* contains plain text and Notes design elements as well as HTML tags, for example <h2> and </h2> for highlighting text.

The action specified for the hotspot is *@Command([Compose]; “New Account”)*. If the user clicks on this hotspot a new document using the form *New Account* is composed.



[<h2>]User Registration Form[</h2>]

All Fields marked with ○ are required

First Name:	○	FirstName
Middle Initial:		MiddleInitial
Last Name:	○	LastName
Company:		Company
Phone Number:		OfficePhoneNumber
E-mail:	○	Email
Password:	○	NewPassword

This picture shows the upper part of the form. As you see, it is used to gather information about the user. Let us have a closer look at each field:

- **FirstName**

  This field is marked as a required field. Therefore, the input translation formula is coded as *@Trim(FirstName)* which removes any leading, trailing, and redundant spaces from the input string.

  The input validation formula is coded as:

```
@If(
  @Length(FirstName) < 2; @Failure("First name is a required field. Please go
back to the form to include a first name.");
  @Contains(FirstName;
@Explode("<x>x{x}x[x]x?x!x#x-x=x+x*x(x)x&x^x%x$x@x/x \ \ x \ ""; "x"));
@Failure("Please use only letters in your first name");
  @Success)
```

First, this formula checks if the user name entered consists of more than 1 character. If only 1 character is entered as first name an error message is issued. The next check looks for unwanted characters in the name field. This is done by converting a string of special characters to a text list using

```
@Explode("<x>x{x}x[x]x?x!x#x-x=x+x*x(x)x&x^x%x$x@x/x\ \x\""; "x").
```

In this formula the character *x* serves as delimiter between the single special characters which are to be in the resulting text list. *@Contains* then determines whether one of the special characters is contained in the input string or not. Depending on the result of this check an error message is issued or *@Success* is called to indicate a successful completion of the tests.

- **MiddleInitial**

  For this field an input translation formula *@Trim(MiddleInitial)* is used and an input validation formula checks for special characters.

- **LastName**

  The checks for this field are the same as for *FirstName.*

- **Company**

  Only an *@Trim(Company)* is coded as input translation formula for this field.

- **OfficePhoneNumber**

  No checks are implemented for this field.

- **E-mail**

  Again, as input translation formula an *@Trim(Email)* is coded, and as input validation formula we use

```
domain := @RightBack(Email; ".");
@If(
   @Trim(Email) = "" |
   !@Contains(Email; "@") |
   @Length(domain) < 2 | @Length(domain) > 3 |
   (@Length(domain) = 3 & @Member(@LowerCase(domain);
"com":"edu":"org":"net":"gov":"mil":"int") = 0) |
   @Trim(@Left(Email; "@")) = "";
   @Failure("A valid Internet E-mail address is required. Please go back to the
form to enter or correct the E-mail address.");
  @Success)
```

This formula uses a temporary variable *domain* which holds the domain of the entered E-mail address. The test completes successfully if the following conditions are met:

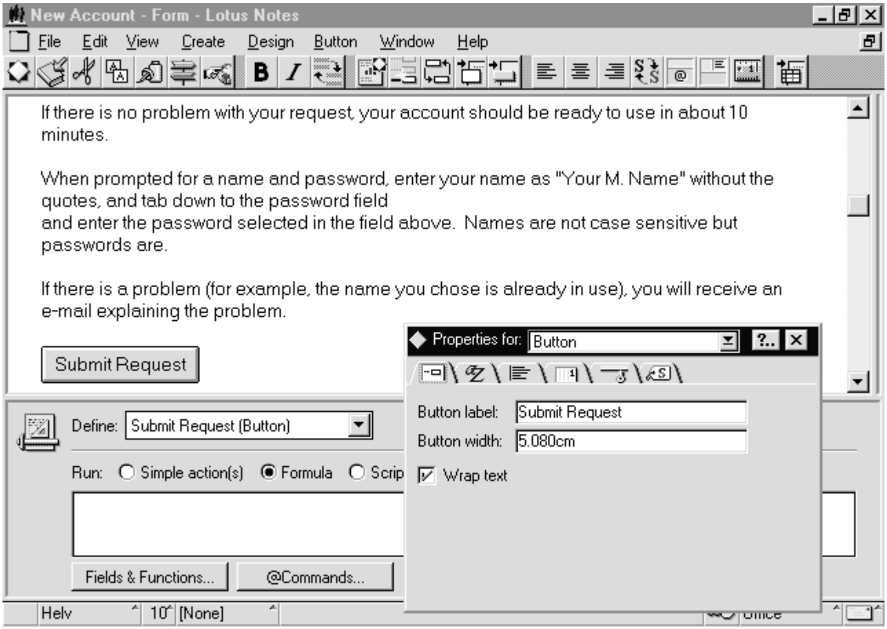
- *Email* must not be blank.
- *Email* must contain the character @.
- The number of characters of domain must be two or three.
- If domain consists of three characters it must be one of *com*, *edu*, *org*, *net*, *gov*, *mil*, or *int*.
- The part of *Email* left of @ must not be blank.

If one of these conditions is not met *@Failure* is used to issue the error message.

- **NewPassword**

  This is a shared field. The definition of this field contains a default value formula which builds a new password randomly. Input translation and validation formulas are used to check the contents of this field.

The second part of the *New Account* form looks like this:



If there is no problem with your request, your account should be ready to use in about 10 minutes.

When prompted for a name and password, enter your name as "Your M. Name" without the quotes, and tab down to the password field
and enter the password selected in the field above. Names are not case sensitive but passwords are.

If there is a problem (for example, the name you chose is already in use), you will receive an e-mail explaining the problem.

Submit Request

Define: Submit Request (Button)

Run: ○ Simple action(s) ● Formula ○ Scrip

Fields & Functions... @Commands...

Helv 10 [None]

Properties for: Button

Button label: Submit Request

Button width: 5.080cm

☑ Wrap text

As you can see there are some explanations for how to enter the user ID and password combination in future sessions. There is also a button to submit the request to the Notes server.

If you are in design mode and scroll down you will find hidden fields in this form. These fields are used to store information about this user record during the process of authorizing the user to access the Notes server. We will discuss these fields later on when we use them.

Error Handling on Form *New Account*

Keep in mind that we are discussing a Domino application which displays its forms using a Web browser. Therefore, the error handling differs from native Notes applications.

You may specify your error message string using the *@Failure* formula if you detect invalid input. Domino displays this message in its own HTML page which is generated on-the-fly. Of course, the string you pass to the *@Failure* formula may contain HTML tags to control the formatting of the failure message. For example, *@Failure("<h2>Invalid Input</h2>")* would generate a page with "Invalid Input" in big bold letters.

If your tests result in an *@Success* formula, you may add a field to your form called *$$Return*. The formula in this field gets evaluated on *@Success*, and the resulting text is displayed on an HTML page. As with *@Failure*, the text string can include HTML tags. In our application the following string is used as a value for the field *$$Return*.

"<body>"+

"<p>Thank you, " + FirstName + ", your request will be processed." +

"<p>If there is no problem with your request, your account should be ready to use in about 10 minutes." +

"<p>Your user name will be <b>" + FullName + "</b>." +

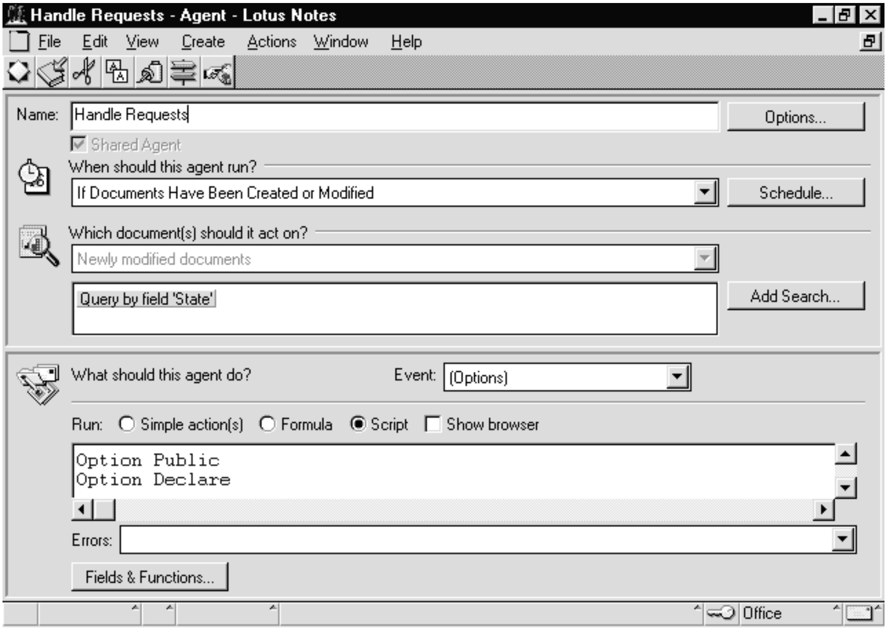
"<p>If there is a problem (for example, the name \ "" + FullName + " \ " is already in use), you will receive an e-mail explaining the problem." +

"<p><a href= \ "http://domino.lotus.com \ ">Lotus Domino Home Page</a>"

Note *FullName* is a hidden field on the form which is composed using the fields *FirstName, MiddleInitial*, and *LastName.*

Processing User Requests

Once a user has filled out a request form and submitted it for processing, a new document is created in the database. Every time a document is created or modified a Notes agent called *Handle Requests* runs to process these documents.



Name: Handle Requests

☑ Shared Agent

When should this agent run? If Documents Have Been Created or Modified

Which document(s) should it act on? Newly modified documents

Query by field 'State'

What should this agent do? Event: (Options)

Run: ○ Simple action(s) ○ Formula ● Script ☐ Show browser

```
Option Public
Option Declare
```

Errors:

This agent acts on all new or modified documents which contain a field named *State* and the content of this field is “P,” for Pending. *State* is one of the hidden fields in form *New Account*, and its initial value is “P.” Therefore, this agent handles all documents which have been submitted by users to get an account on the Notes server. The actions the agent is to perform are coded in LotusScript.

Here is the main routine of this agent (declarations and settings of constant strings are done in the declaration section of the agent):

```
Sub Initialize

  Set s = New NotesSession
  Set db = s.CurrentDatabase

  '** Set Name & Address book where new accounts
  '** should be added
  Set nabPeople = New NotesDatabase( "", nabPeoplePath$ )
  Set nabGroups = New NotesDatabase( "", nabGroupsPath$ )
  '** Get the collection of all documents this agent should
  '** run against
  '** (Those that have been newly modified or created)

  Dim coll As NotesDocumentCollection
  Set coll = db.UnprocessedDocuments

  Dim doc As NotesDocument
  Dim i As Integer

  '** Loop over the documents to handle.

  For i = 1 To coll.Count
    Set doc = coll.GetNthDocument( i )

    '** Handle Change Password Requests
    If (doc.Form(0) = "ChPw" And doc.State(0) = "P") Then
      Call HandleChangePassword( doc )
    End If

    '** Handle New Account Requests
    If (doc.Form(0) = "NewAcc" And doc.State(0) = "P") Then
      Call HandleNewAccount( doc )
    End If

    '** UpdateProcessedDoc makes sure the agent won't run on
    '** this document again.

    Call s.UpdateProcessedDoc( doc )
  Next

End Sub
```

At the beginning the variables *s, db,* and *coll* are set to have access to the Notes session, the database, and the document collection which contains all unprocessed documents. Using a loop, each unprocessed document is accessed and processed depending on the contents of the fields *Form* and *State*. If a document contains a request for a new account the subroutine *HandleNewAccount* is called. Input parameter to this subroutine is the document to be processed (variable *req*).

```
Sub HandleNewAccount ( req As NotesDocument )
  Dim status As String
  On Error Goto Oops

  Call WriteInitialAgentData( req )
```

```
'** Password must have been hashed.
If Not IsPasswordOK( req.NewPassword(0) ) Then
  req.AgentStatus = "Password ill-formed"
  Goto Done
End If

'** Check that the user doesn't already exist in the NAB
Dim docPerson As NotesDocument
Set docPerson = GetPersonDocument( req.FullName(0) )

If Not (docPerson Is Nothing) Then
  req.AgentStatus = "Duplicate name"
  req.SendMailTo = req.Email
  Dim message As String
  message = ........
  req.Message = message
  Goto Done
End If

'** Create the person document in the NAB


Set docPerson = New NotesDocument( nabPeople )

docPerson.Form = "Person"
docPerson.Type = "Person"
docPerson.LastName = req.LastName
docPerson.FirstName = req.FirstName
docPerson.MiddleInitial = req.MiddleInitial
docPerson.FullName = req.FullName
docPerson.OfficePhoneNumber = req.OfficePhoneNumber
docPerson.HTTPPassword = req.NewPassword
Call docPerson.ComputeWithForm( False, False )
Call docPerson.Save( False, True )

'** Add the person to the proper groups

Forall group In req.GroupsToJoin
  Call AddUserToGroup( req.FullName(0), group )
End Forall

'** Create the person document in this database

Set docPerson = New NotesDocument( req.ParentDatabase )

docPerson.Form = "Person"
docPerson.LastName = req.LastName
docPerson.FirstName = req.FirstName
docPerson.MiddleInitial = req.MiddleInitial
```

```
  docPerson.FullName = req.FullName
  docPerson.OfficePhoneNumber = req.OfficePhoneNumber
  docPerson.HTTPPassword = req.NewPassword
  docPerson.Email = req.Email
  docPerson.Company = req.Company
  Call docPerson.ComputeWithForm( False, False )
  Call docPerson.Save( False, True )

  req.AgentStatus = "Successful"
  req.SendMailTo = ""

Done:
  On Error Goto Bail

  Call WriteFinalAgentData( req )

  Exit Sub

Oops:
  status$ = "Error " & Err() & " at line " &  _
Erl() & ": " & Error()
  req.AgentStatus = status$
  Resume Done
  Bail:
  status$ = "Error " & Err() & " at line " & _
            Erl() & ": " & Error()
  Resume BailOut
BailOut:

End Sub
```

First, this subroutine calls *WriteInitialAgentData* which is a subroutine to set initial bookkeeping information into the request document, for example it sets the field *AgentStatus* to “Agent running”. Next it calls a subroutine to check if the password is built correctly. The following check uses the function *GetPersonDocument* to look up the Name and Address Book for an entry of the requesting person. If such an entry is found the following error message is composed:

"The name " & req.FullName(0) & " is already in use. Please try registering again with a different name. If you have registered more than once, the first attempt may have succeeded, and the subsequent attempts are duplicates. Please try your account before re-registering if this might be the case.

If such an entry is not available in the Name and Address Book, the data of the request document are used to build a new entry in the Name and Address Book of the server, to add the user to a group, and to create a document using the form *Person* in the database. For housekeeping reasons the field *AgentStatus* in the request form is set to “Successful,” and routine *WriteFinalAgentData* sets field *State* in the request document to “D,” for Done.

Sending Error Messages to Web Users

During the processing phase of the request documents the field *SendMailTo* is set either to the E-mail address entered on the request form or to the E-mail address of the owner of the Notes server. If the authorization process ends without an error the field *SendMailTo* is cleared. Furthermore, there is another hidden field on the request form called *MailSent* which has the initial value “N,” for Not sent. A second agent called *Send Problem Mail* runs on new or modified documents. It picks up all documents which meet the following condition:

```
doc.HasItem( "SendMailTo" ) And _
doc.SendMailTo(0) <> "" And _
 doc.MailSent(0) = "N"
```

For each of the documents the subroutine *SendProblemMail* is carried out.

```
Sub SendProblemMail ( req As NotesDocument )
  Dim email As NotesDocument

  Set email = New NotesDocument( db )
  email.Form = "Memo"
  email.SendTo = req.SendMailTo
  email.Subject = "Problem with request: " &
req.AgentStatus(0)
  email.SaveMessageOnSend = False

  Dim bodyItem As NotesRichTextItem
  Set bodyItem = email.CreateRichTextItem( "Body" )
  Call bodyItem.AppendText("There was a problem
with your " & _
       req.Form(0) & " request: " &
req.AgentStatus(0) & "." )
  Call bodyItem.AddNewLine( 2 )
  If req.Message(0) <> "" Then
    Call bodyItem.AppendText( req.Message(0) )
    Call bodyItem.AddNewLine( 2 )
  End If
  Call bodyItem.AppendText("If you need help, " & _
       "send mail to " &  helpEmail$ & _
       ". Include the text of this message, " & _
       "and the problems you are experiencing." )
  Call bodyItem.AddNewLine( 1 )

  email.Send( False )

  req.MailSent = "Y"
  Call req.Save( False, True )

End Sub
```

Handling Password Change Requests

To let a user change the password over the Web, a similar concept is used. In this case a request document is created which causes the agent *Handle Requests* to branch to subroutine *HandleChangePassword.* This subroutine performs the update of the Name and Address Book.

Summary

This section discussed a sample of how to register Web users for Domino-based applications where users can create accounts for themselves.

Caution This sample is designed to allow anyone with Web connectivity to your server to create a user name and password for themselves on your server. This will not allow them to use Notes to access your data, but it will allow them to use their Web browser through Domino to access and possibly change data. Be very careful in deploying this application or modified versions of it. This application modifies your Public Name and Address Book, which could, in certain circumstances, compromise security.

Caution If many of your application ACLs are set to default reader access, any Web browser will also be able to access them. Protect confidential databases from Web users by setting the Anonymous user to No Access.

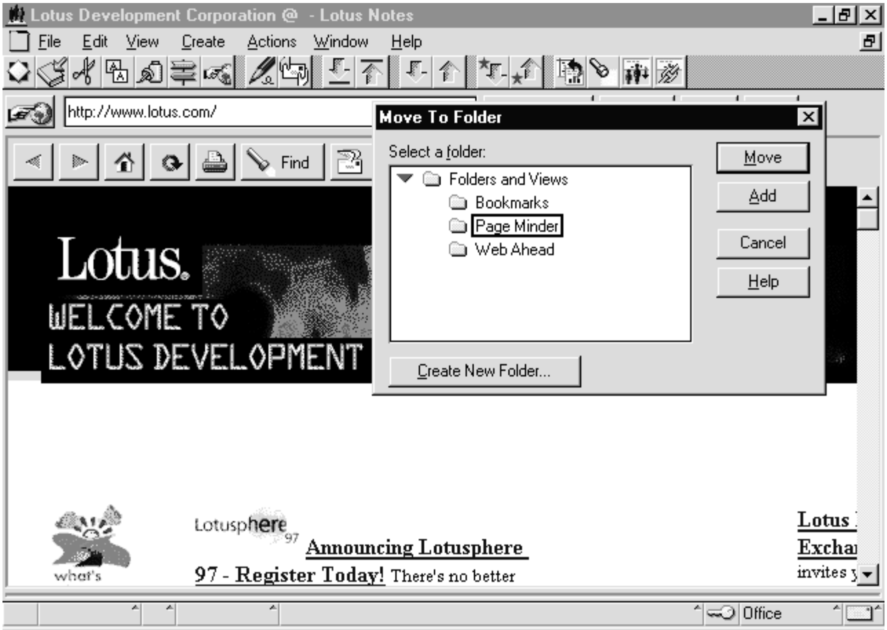
Personal Agents — The Page Minder Agent

Today the World Wide Web provides you with abundant sources of information. While browsing the Web you will certainly find Web sites which are of interest to you. In order to be informed about the latest status on these pages you have to retrieve them now and then to check if they have been updated. This task can be very time consuming. To save time you can use the Page Minder agent included in the Personal Web database, to do the job automatically.

Using the Web Navigator Database

Note The following example assumes that you have selected Notes as your Web browser in your location document.

When you open a URL, Notes retrieves this URL and displays the Web page. All Web pages you have accessed are listed in the view *All Documents*. If you are interested in a specific page and you want Notes to check whether this page has been updated or not, you should put this Web page into the folder *Page Minder*.

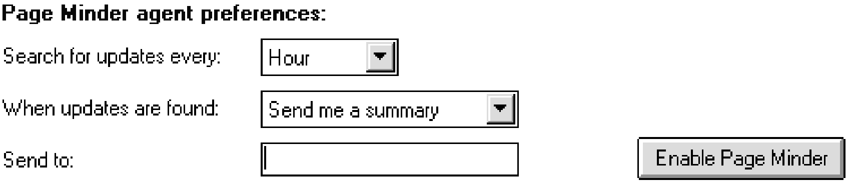


All documents in this folder will be processed by the Page Minder agent.

The Settings for the Page Minder Agent

The purpose of this agent is to monitor specific Web pages and inform the current user or whoever is listed in the Internet Profile by sending a newsletter summary or the changed Web page.

To see how this agent works let us look at the parts of the Internet Profile, stored in your Personal Web database, which are related to the Page Minder agent:



Page Minder agent preferences:

Search for updates every: Hour

When updates are found: Send me a summary

Send to:

In the field labeled *Search for updates every:* you specify the frequency for this agent, this means how often this agent will run. You may have the agent scheduled every hour, every 4 hours, once a day, or once a week. The field *When updates are found:* is used to tell the agent in which way it should send the information to the user whose Notes E-mail address you specify in the field *Send to:* — send a summary or the whole page. Once you have made your choices, click on the button *Enable Page Minder* and save the Internet Profile.

Caution If you have chosen to retrieve Web pages from your workstation don't forget to enable agents to run on your workstation in the File - Tools - User Preferences menu.

The Page Minder Agent

The following section will look in more detail at the implementation of the Page Minder agent. The agent is written in LotusScript. The code is split into the following parts:

- **Declarations**

  Here you find the declaration statements for globally used variables.

- **Initialize**

  This is the main routine of the agent. Notes will transfer control to this part of the LotusScript program when it invokes the agent.

- **LocationOK**

  This function verifies that the current locations settings are correct for local Web retrievals.

- **NeedToRun**

  This routine checks whether the agent should be executed or not.

- **MindURLs**

  This subroutine performs the test on each document in the page minder folder to see if the document has changed.

- **HasDocumentChanged**

  This part contains a function which is used by subroutine *MindURLs*. It determines if a specific Web page has changed.

- **RetrieveOverInternet**

  This is the subroutine which contains the LotusScript statements to retrieve either the Last Modified date from the HTTP header of a Web page or the complete Web page.

Let us step through the code of this agent and look at some important parts.

Tip Print out the Agent script using the File-Export menu while editing the agent so you can have the code to look at as we describe the functionality.

Initialize

This part is used to set up the working environment for the agent. The variables to hold the information about the Notes session, the agent's saved data document, the database, and the agent itself are initialized at the beginning. Next, the test to see if the current location is set up correctly (Function *LocationOK*) is performed. If this test is passed successfully, there are checks to see if the Web database is available, if the Page Minder folder is accessible, if there are documents in the Page Minder folder, and if there is an Internet Profile document in the database. If these tests are passed successfully, the agent's run frequency, the e-mail address, and the method of sending the results are retrieved. If the function *NeedToRun* indicates that the agent should be executed the subroutine *MindURLs* is called.

LocationOK

First, all address books are scanned to find the private one which must contain information about the Internet settings of the current location. If an entry for the current location is found checks are performed to see if Notes is selected as Web browser and if the location is set up for local Web retrievals.

NeedToRun

This function parses parameters containing the time-stamp detailing the time the agent was run the last time and the frequency as specified in the Internet profile. Since changes to the frequency in the Internet profile do not affect the schedule settings in the agent's design, Notes will schedule this agent every 30 minutes. It is the up to the agent to determine if it is time to process the Web pages or not. Therefore, the return value of this function indicates whether the agent continues or not.

MindURLs

Depending on the parameter this subroutine will send a summary of all changed Web pages to the Notes user or it will send the changed Web pages. For each document in the Page Minder folder there are tests for the existence of URL information in the document and if the document has changed. If the function *HasDocumentChanged* returns *True* the appropriate Web page is retrieved from the Web and put into the folder *Page Minder*. Next, the document is sent to the Notes user or the URL of the retrieved document is added to the summary list.

HasDocumentChanged

The purpose of this function is to determine if a Web page has changed. To determine this, the Last-Modified date of the Web page is retrieved from the HTTP server.

RetrieveOverInternet

This function handles all calls to the Internet. It either gets the Last-Modified information or the Web page. It also covers the different access methods to the Web, for example using a proxy.

If this agent is enabled, it will run periodically on the workstation and keep all documents in the Page Minder folder up-to-date.

Case Study: Millennia Multimedia

The following case study makes use of Domino's document publishing, Web user interaction and search capabilities.

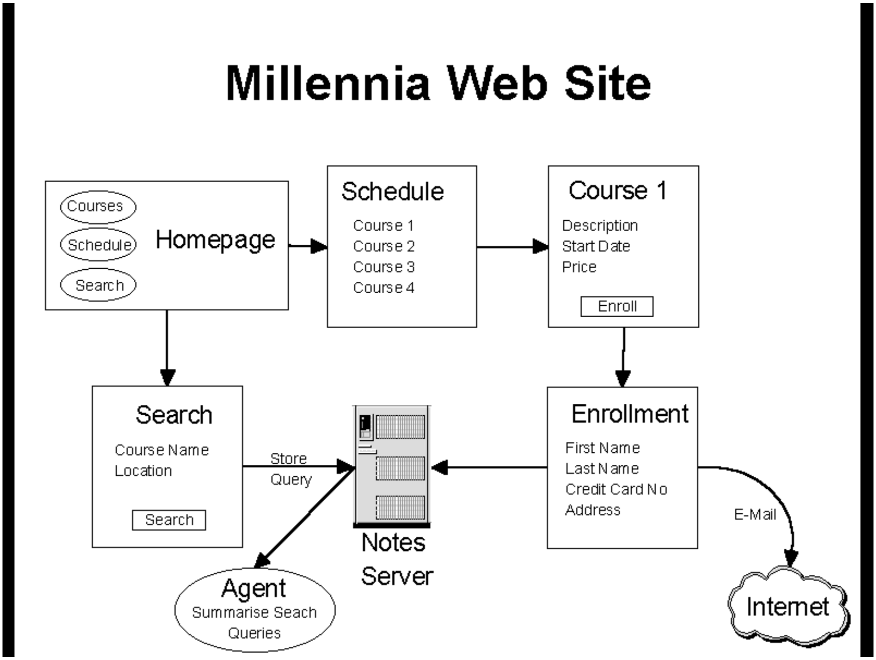
Millennia Multimedia wishes to publish details of their courses onto the Internet using the Notes HTTP server. Currently a lot of their course information is held in an Oracle database that they use to produce billing information. They would like to be able to still use the Oracle database but leverage the power of Notes to attract new customers from the Internet. Additionally, they would like to be able to offer visitors to the Web site the ability to register for a course automatically.

NotesPump

NotesPump is used to populate a Notes database with course information from the Oracle database and to take values from the Web course enrollment forms back into Oracle. For a description on how NotesPump was set up to do these tasks, see the NotesPump chapter in this book. For the purpose of this case study we will only look at data that is stored within Notes.

The diagram below shows from a top level view how users visiting the Web site will be able to enroll in a course. Starting from the home page they will be able to view the current course schedule and look at each one to determine which would be suitable to them. When the users click a button on the form used to enroll in a particular course, they are presented with a form that they must complete with personal information such as their name and address, how they would like to pay, their occupation etc. The form is stored in Notes and passed back into Oracle using NotesPump. After Oracle processes each new form a confirmation note is sent to the users' E-mail address detailing their course enrollment information.

The user is also able to search the site for a particular course by using a search form. These search queries are stored as documents in Notes and are used by an agent running on the Notes server to provide statistics on the number of searches per course.



The Start

First, you need to create a new database which will contain the pages you are going to display onto the Internet.

1. On your desktop, choose File - Database - New or **CTRL-N**.
2. Select the server where the database will reside, type in a title and a database name.
3. Select the OK button.

The home page

The start of any Web site is the home page. This is the default page that is displayed when a person first enters a Web site.

The home page for our case study was created using a graphics program and then imported into a Notes Navigator as a background bitmap. As the Notes HTTP server can only display the background bitmap, all the text, graphics and lines need to be created entirely on the picture.

Once the picture is complete, open the database you created in the steps above.

1. Select Create - Design - Navigator from the menu bar.
2. Open your home page bitmap in your paint program and copy it to the clipboard. (This will differ depending on the paint program you are using.)
3. Select Create - Graphic Background from the menu. Your home page bitmap should appear in Notes.

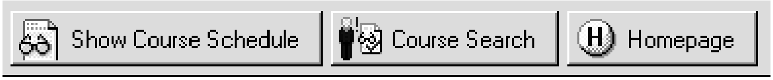
   Note If the colors of your bitmap do not appear correctly, most likely it is because you have used more than 256 colors.
4. In the areas where you have decided to have buttons or links to other parts of the Web site / database, you need to add a hotspot rectangle by selecting Create - Hotspot Rectangle from the menu and dragging it over the area you wish to designate as a hotspot.
5. Give the new hotspot a meaningful name, such as LinkToCourses.
6. Select an action for the hotspot. If you are linking to a view or another navigator you can use the Simple Actions to create the link, or if you need to do something more advanced, click the Formula or Script buttons and type in the code. For example, in our example the Search Site hotspot needs to create a new document, so the formula entered for the hotspot is @Command([Compose]; "SearchSite").
7. Repeat these steps for each hotspot that you need to create on your navigator.
8. When you are ready, select File - Save from the menu.

Course View

This view is used to display a list of course titles that the company offers. Selecting one of the courses will give the user a description of the course contents.

This is a simple view containing two columns, the first the course name and the second the course description.

At the top of the view are three action buttons.



1. Show Course Schedule. This button links the user to the Course Schedule view that shows details on when and where a course is being held. The action button formula is @Command([OpenView]; "ClassSchedule")

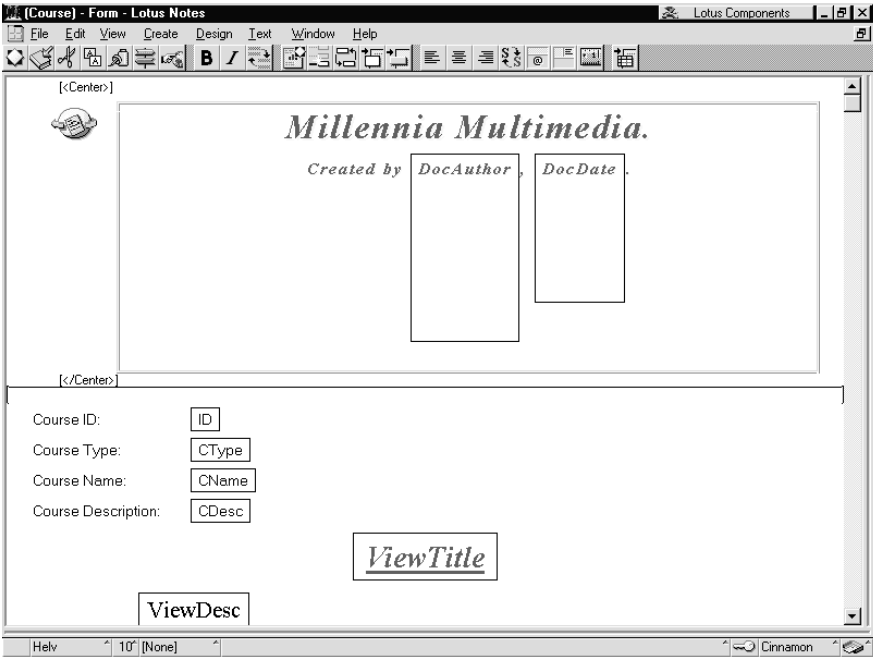
2. Course Search. This button links the user to the Search Site form that will allow them to search for when a particular course is being held in a city. The action button formula is @Command([Compose]; "SearchSite")

3. Homepage. This button takes the user back to the home page so that they can select another option.

   Tip It is important to give the user a way back to a familiar location from every page that they see.

Course Form

The class documents are created directly from the Oracle database using NotesPump.



(Course) - Form - Lotus Notes
Lotus Components
File Edit View Create Design Text Window Help
B I
Helv 10 [None]
Cinnamon

[<Center>]

Millennia Multimedia.

Created by DocAuthor, DocDate.

[</Center>]

Course ID: ID

Course Type: CType

Course Name: CName

Course Description: CDesc

ViewTitle

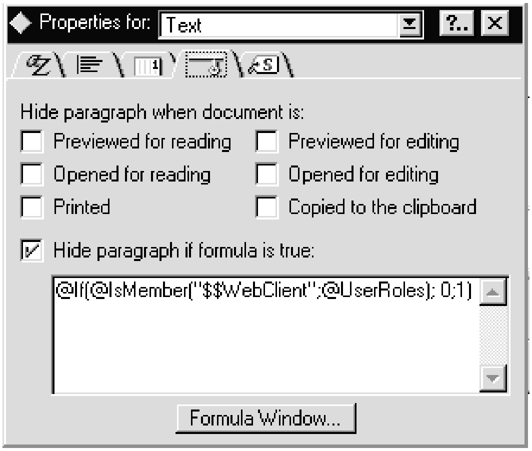
ViewDesc

In the top half of the screen is a subform containing the document title that will be added to the top of all the forms. Notice that there are some embedded HTML tags in the banner to make it appear central when viewed using a Web browser.

The commands to center a paragraph on a page are <Center> *your text* </Center>.

Note If you do not put the text in a style named HTML you must enclose the whole string in square brackets, for example, [<Center> *your text* </Center>]

These HTML codes now cause us a problem when we are viewing the database using a Notes client as we will be able to see them as plain text. What we can do is to look at the user roles of the database ACL to determine what type of client is viewing the database and use a Hide-when formula to hide the text if the form is being viewed from Notes.



Hide paragraph when document is:

□ Previewed for reading □ Previewed for editing
□ Opened for reading □ Opened for editing
□ Printed □ Copied to the clipboard

☑ Hide paragraph if formula is true:

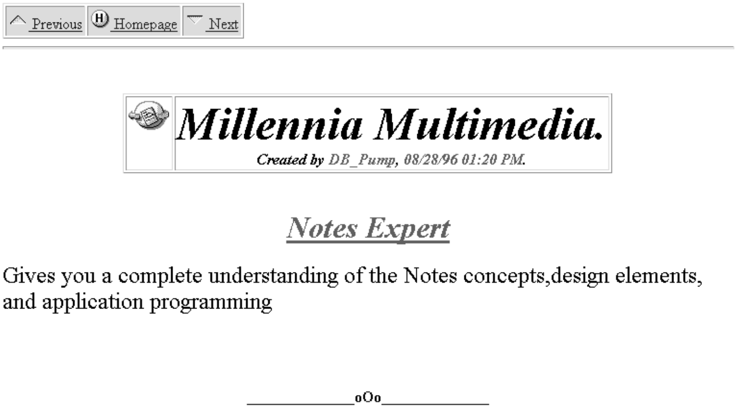
```
@If(@IsMember("$$WebClient";@UserRoles); 0;1)
```

This formula sets a value of 1 or 0, (true or false), depending on whether it finds the current user is a member of $$WebClient in the database's UserRoles field.

Tip Use this Hide-when formula to hide HTML tags from a Notes client. @If(@IsMember("$$WebClient";@UserRoles); 0;1).

The top four fields on the form describe what the course is about with some additional fields that are key fields to the Oracle database. When you display this information to a Web user you do not want to present them with all the information here so we create two new fields, ViewTitle and ViewDesc that are computed when displayed fields.

This is how the page looks from a Web client.



Millennia Multimedia.

Created by DB_Pump, 08/28/96 01:20 PM.

Notes Expert

Gives you a complete understanding of the Notes concepts,design elements, and application programming

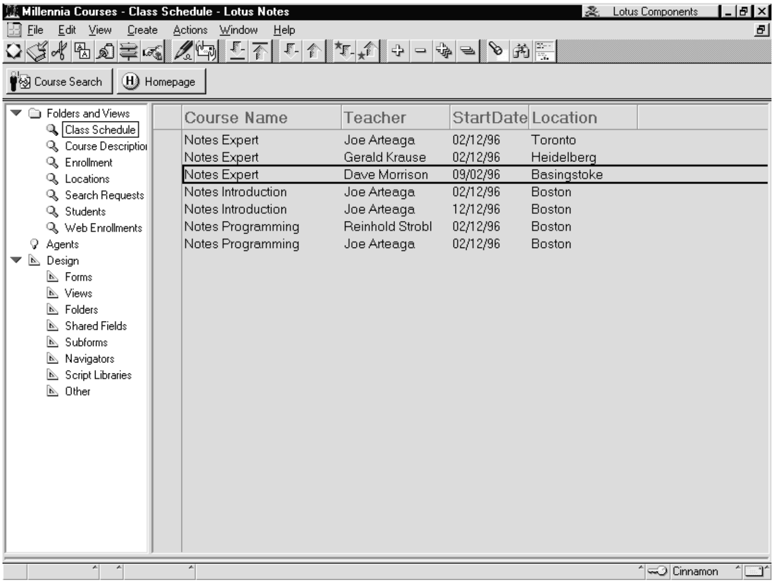
___________oOo___________

At the top of the page are three action buttons.

1. Previous. This button will take the user to the previous document in the view. If there are no more documents then the view is re-displayed.
2. Homepage. This button takes the user back to the home page.
3. Next. This button will take the user to the next document in the view. If there are no more documents then the view is re-displayed.

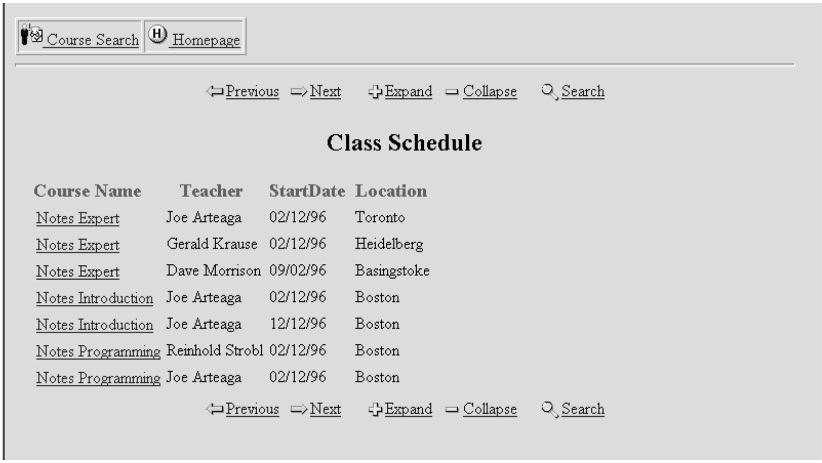
Class View

The class view contains all the information about when and where a course is being held.



Course Name	Teacher	StartDate	Location
Notes Expert	Joe Arteaga	02/12/96	Toronto
Notes Expert	Gerald Krause	02/12/96	Heidelberg
Notes Expert	Dave Morrison	09/02/96	Basingstoke
Notes Introduction	Joe Arteaga	02/12/96	Boston
Notes Introduction	Joe Arteaga	12/12/96	Boston
Notes Programming	Reinhold Strobl	02/12/96	Boston
Notes Programming	Joe Arteaga	02/12/96	Boston

This is how the view is displayed when using a Web client.



Class Schedule

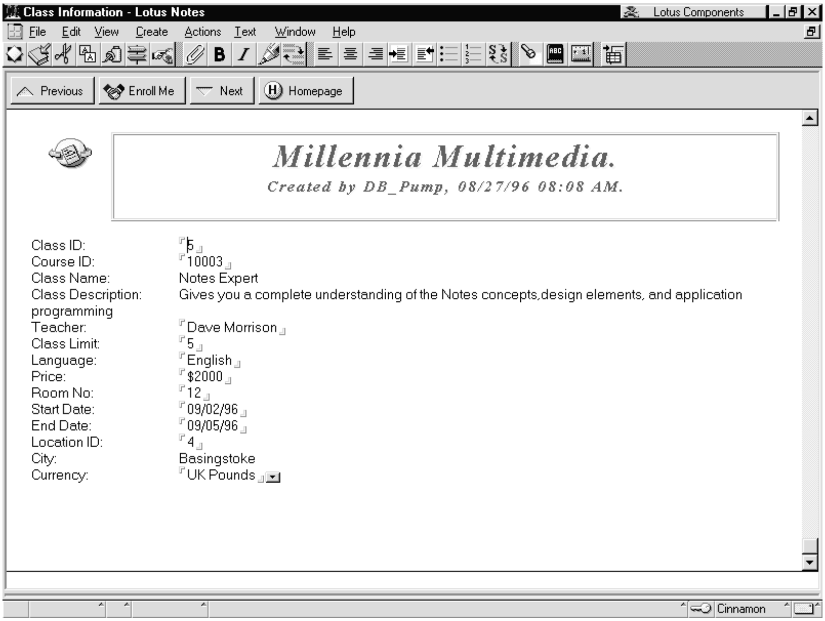
Course Name	Teacher	StartDate	Location
Notes Expert	Joe Arteaga	02/12/96	Toronto
Notes Expert	Gerald Krause	02/12/96	Heidelberg
Notes Expert	Dave Morrison	09/02/96	Basingstoke
Notes Introduction	Joe Arteaga	02/12/96	Boston
Notes Introduction	Joe Arteaga	12/12/96	Boston
Notes Programming	Reinhold Strobl	02/12/96	Boston
Notes Programming	Joe Arteaga	02/12/96	Boston

Note Background view colors, column heading colors and font sizes are translated when viewing through a Web client.

Clicking on one of the links will take you to the Class Form.

Class Form

The class form displays to the user all the details for a particular course: when it is being held, who is running it, where it is being held, how much it is, etc.

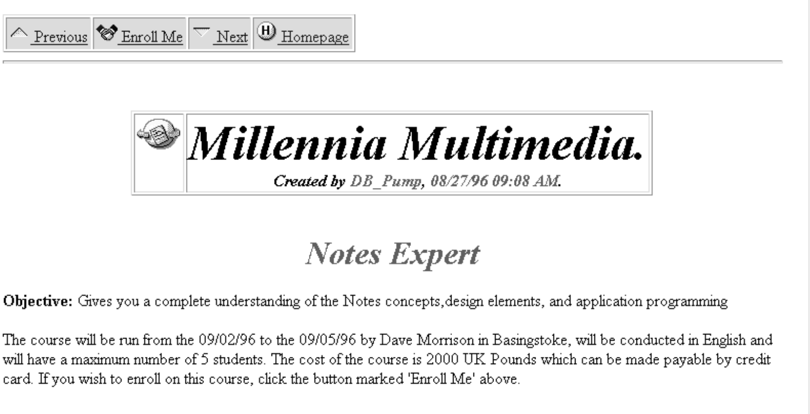


Millennia Multimedia.

Created by DB_Pump, 08/27/96 08:08 AM.

Class ID:	5
Course ID:	10003
Class Name:	Notes Expert
Class Description:	Gives you a complete understanding of the Notes concepts,design elements, and application programming
Teacher:	Dave Morrison
Class Limit:	5
Language:	English
Price:	$2000
Room No:	12
Start Date:	09/02/96
End Date:	09/05/96
Location ID:	4
City:	Basingstoke
Currency:	UK Pounds

The above figure shows how the form looks when it is displayed in edit mode. When the form is opened in read mode we use two additional fields that are set to be computed when composed to display the same information in a friendlier manner. The first field simply displays the name of the course, and the second displays a text paragraph made up from the various fields on the form. As a Web client will be viewing this form in read only mode, this is what they see.



Millennia Multimedia.

Created by DB_Pump, 08/27/96 09:08 AM.

Notes Expert

Objective: Gives you a complete understanding of the Notes concepts,design elements, and application programming

The course will be run from the 09/02/96 to the 09/05/96 by Dave Morrison in Basingstoke, will be conducted in English and will have a maximum number of 5 students. The cost of the course is 2000 UK Pounds which can be made payable by credit card. If you wish to enroll on this course, click the button marked 'Enroll Me' above.

The formula for the second paragraph is:

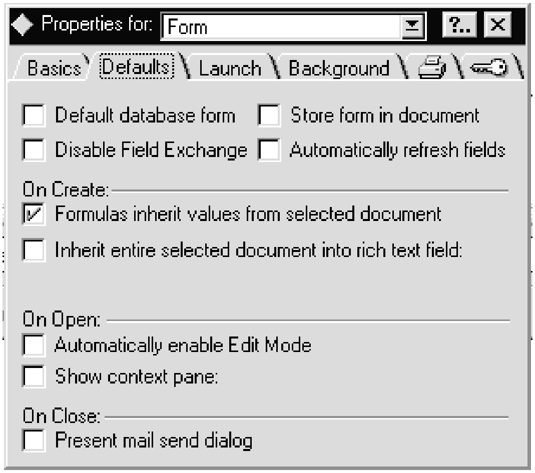
```
@If(@IsNewDoc; ""; "The course will be run from the " +
@Text(StartDate) + " to the " + @Text(EndDate) + " by " +
Teacher + " in " + City + ", will be conducted in " +
Language + " and will have a maximum number of " +
@Text(Limit) + " students. The cost of the course is " +
@Text(Price) + " " + Currency + " which can be made payable
by credit card. If you wish to enroll on this course, click
the button marked \'Enroll Me\' above.")
```

At the top of the form are four action buttons.

1. Previous. This button will take the user to the previous document in the view. If there are no more documents then the view is re-displayed.
2. Enroll Me. This button will compose a new Web Enrollment form for the user to supply personal information so that they can register for a course. The formula for this button is @Command([Compose]; “WebEnroll”);
3. Next. This button will take the user to the next document in the view. If there are no more documents then the view is re-displayed.
4. Homepage. This button takes the user back to the home page.

Web Enrollment Form

This form is used to gather enrollment related information from the user. The only place that this form can be created is from the class schedule form as it needs to inherit several fields from this document.



□ Default database form □ Store form in document

□ Disable Field Exchange □ Automatically refresh fields

On Create:

☑ Formulas inherit values from selected document

□ Inherit entire selected document into rich text field:

On Open:

□ Automatically enable Edit Mode

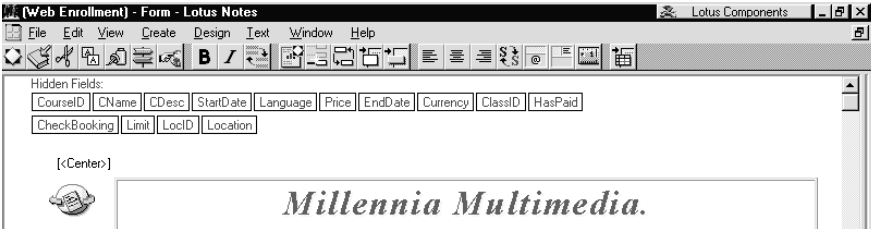
□ Show context pane:

On Close:

□ Present mail send dialog

To enable field inheritance from document to document, check the box shown above in the Forms InfoBox.

If we open the form in design mode, this is what we see:



Hidden Fields:

CourseID CName CDesc StartDate Language Price EndDate Currency ClassID HasPaid

CheckBooking Limit LocID Location

[<Center>]

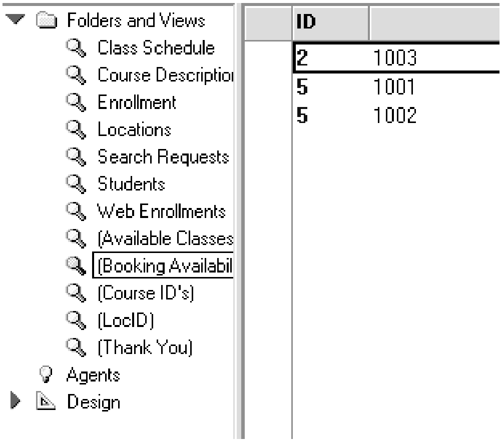
At the top of the form are a number of hidden fields that are mostly inherited from the class schedule form, including two new fields, *CheckBooking* and *HasPaid*.

The *HasPaid* field is used at a later date by the Oracle database for billing purposes.

The *CheckBooking* field is used to determine how many places are left for the course taking into account the number of places already booked. Here is the formula used:

```
@If(@IsError(@DbLookup("":"NoCache"; ""; "(Booking
Available)"; ClassID; 2)); 0; @Elements( @DbLookup(
"":"NoCache"; ""; "Available"; ClassID; 2)))
```

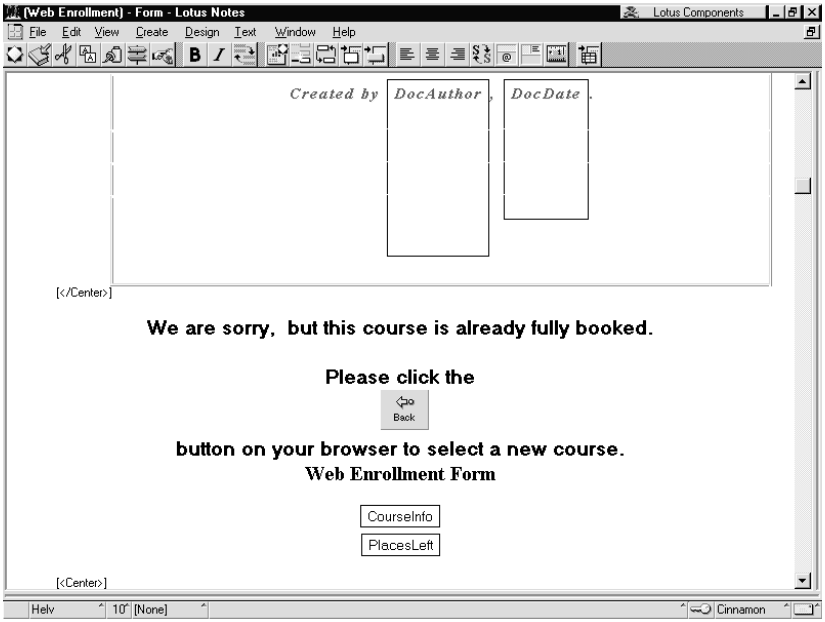
- First the formula checks if there is going to be an error when running this formula. This will occur when no one has yet enrolled in the course. If this is the case, then a default value of zero is set.



ID	
2	1003
5	1001
5	1002

- If there is no error then the next part of the formula repeats the @DbLookUp into a view named *(Booking Available)* (shown above), looking for all the occurrences of the *ClassID* value. This is then converted into a value by using the @Elements command which returns the number of items in a string that have a delimiter such as a comma or colon. In the example above, for course number 5 the @DBLookup will return the string “1001;1002” which is converted into the value 2 by @Elements.
- The result of this formula tells us how many places have already been booked for the course, so far.

Moving down the form we create a paragraph that is displayed to the user if there are no places available for the course.


(Web Enrollment) - Form - Lotus Notes
Lotus Components
File Edit View Create Design Text Window Help
Helv
10
[None]
Cinnamon

Created by DocAuthor, DocDate.

[</Center>]

We are sorry, but this course is already fully booked.

Please click the

Back

button on your browser to select a new course.

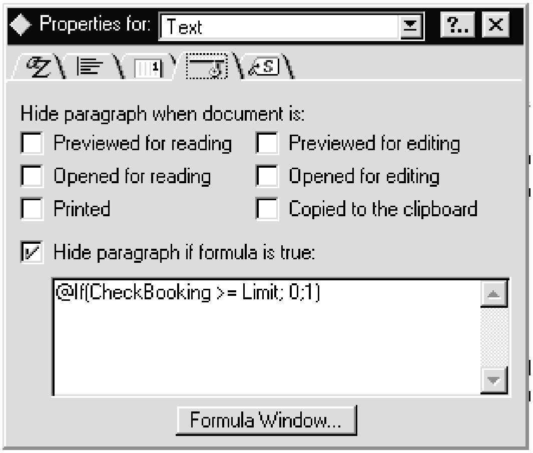
Web Enrollment Form

CourseInfo

PlacesLeft

[<Center>]

This text paragraph has a hide-when formula so that it is only displayed when there are no places available.



Hide paragraph when document is:

□ Previewed for reading □ Previewed for editing

□ Opened for reading □ Opened for editing

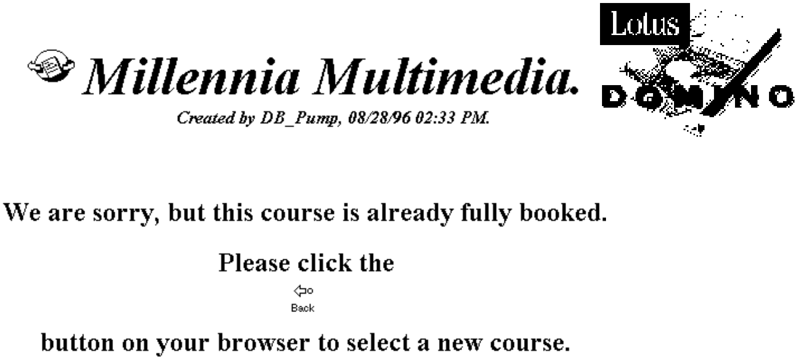
□ Printed □ Copied to the clipboard

☑ Hide paragraph if formula is true:

```
@If(CheckBooking >= Limit; 0;1)
```

This checks to see if the value in the field *CheckBooking* is greater or equal to the value in the *Limit* field and sets a value of 1 or 0, true or false.

The following diagram shows how the form looks when viewed from a Web client if there are no places available for the course.



Millennia Multimedia.

Created by DB_Pump, 08/28/96 02:33 PM.

We are sorry, but this course is already fully booked.

Please click the

button on your browser to select a new course.

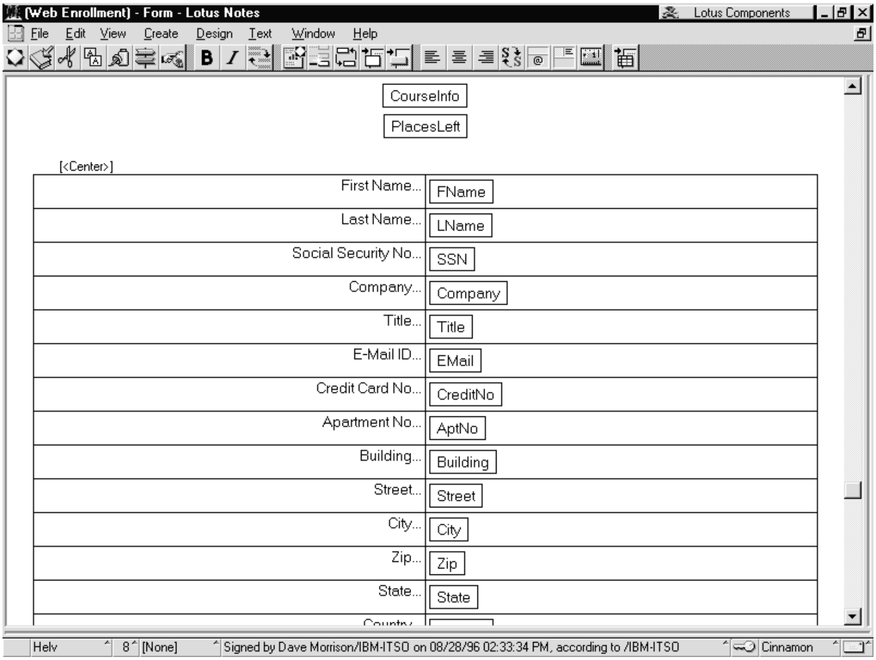
Assuming that the course has places available the remaining fields on the form are displayed. Two of the fields are CourseInfo and PlacesLeft which are informational fields that display a formatted paragraph to the user telling them what they are booking, when the course is being held, etc. The formula for this field is:

```
"Please book me on the " + CName + " course, starting on the
" + @Text(StartDate) + " and ending on the " + @Text(EndDate)
+ ". I understand that if this reservation is confirmed my
credit card will be debited the amount of " + @Text(Price) +
" " + Currency + "."
```

The *PlacesLeft* field directly below this tells the user how many places are available on the course using the formula,

```
"There are currently " + @Text(Limit - checkbooking) + "
provisional places left on this course."
```

Below these two fields are the fields that the user will fill in with information about themselves.


(Web Enrollment) - Form - Lotus Notes
Lotus Components
File Edit View Create Design Text Window Help

CourseInfo

PlacesLeft

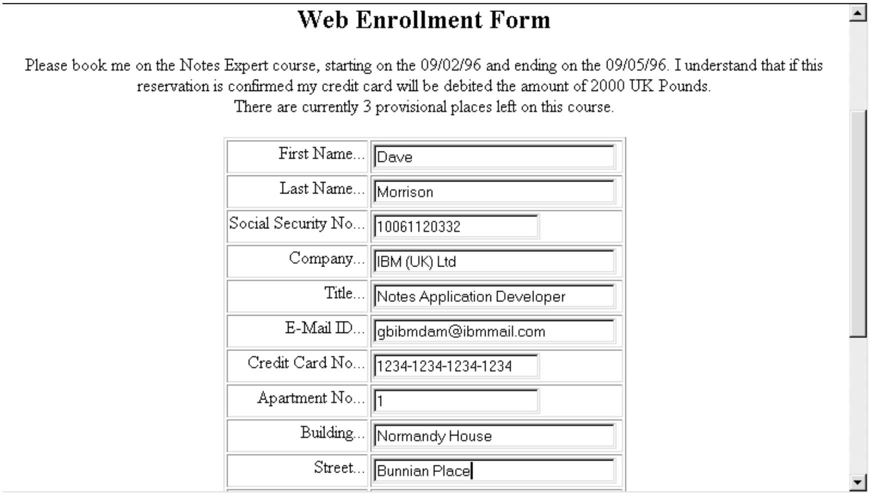
[<Center>]

First Name...	FName
Last Name...	LName
Social Security No...	SSN
Company...	Company
Title...	Title
E-Mail ID...	EMail
Credit Card No...	CreditNo
Apartment No...	AptNo
Building...	Building
Street...	Street
City...	City
Zip...	Zip
State...	State
Country	

Helv 8 [None] Signed by Dave Morrison/IBM-ITSO on 08/28/96 02:33:34 PM, according to /IBM-ITSO Cinnamon

You will notice that each of the fields have been placed in a table, and that the first column has been aligned to the right. When the form is displayed on a Web client the fields are aligned correctly.

This is how the document looks when displayed with the table.

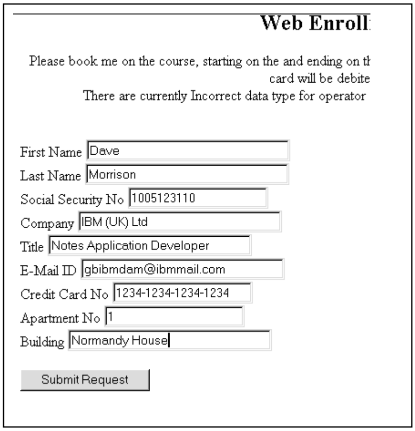


Web Enrollment Form

Please book me on the Notes Expert course, starting on the 09/02/96 and ending on the 09/05/96. I understand that if this reservation is confirmed my credit card will be debited the amount of 2000 UK Pounds.
There are currently 3 provisional places left on this course.

First Name...	Dave
Last Name...	Morrison
Social Security No...	10061120332
Company...	IBM (UK) Ltd
Title...	Notes Application Developer
E-Mail ID...	gbibmdam@ibmmail.com
Credit Card No...	1234-1234-1234-1234
Apartment No...	1
Building...	Normandy House
Street...	Bunnian Place

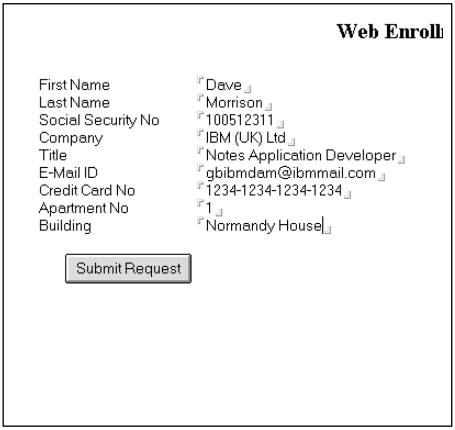
This is the document without the table with a Web client and with Notes.



Web Enroll

Please book me on the course, starting on the and ending on th
card will be debite
There are currently Incorrect data type for operator

First Name: Dave
Last Name: Morrison
Social Security No: 1005123110
Company: IBM (UK) Ltd
Title: Notes Application Developer
E-Mail ID: gbibmdam@ibmmail.com
Credit Card No: 1234-1234-1234-1234
Apartment No: 1
Building: Normandy House



Web Enroll

First Name: Dave
Last Name: Morrison
Social Security No: 100512311
Company: IBM (UK) Ltd
Title: Notes Application Developer
E-Mail ID: gbibmdam@ibmmail.com
Credit Card No: 1234-1234-1234-1234
Apartment No: 1
Building: Normandy House

This happens because tabs are not converted into spaces by the Notes HTTP server.

Tip You can hide the table by removing all the lines from just the top left cell.

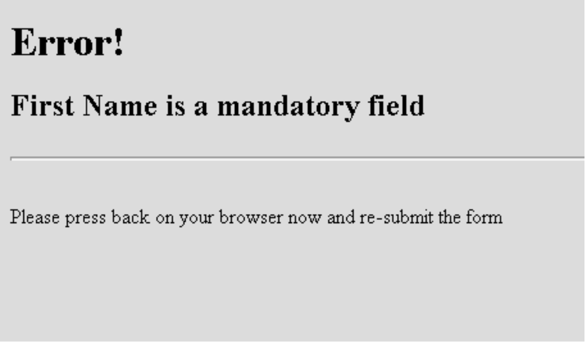
Field Validation

Each field can provide error checking as you would be able to within Notes by typing a formula into the Input Validation area. If a field fails the validation within the formula when using a Web client, the @Failure command is converted into a new Web page with the error text being displayed.

For example, take the field FName which the user must type in their first name. It contains the following Input Validation formula.

```
@If(@Trim(FName) = ""; @Failure("<H1>Error!</H1><H2>First
Name is a mandatory field</H2><HR><BR><P>Please press back on
your browser now and re-submit the form");
@Success)
```

This formula will return an error if the field value is blank. To make this error more presentable to a Web client it includes some HTML formatting codes. Here is the error code displayed using a Web client.



Error!

First Name is a mandatory field

Please press back on your browser now and re-submit the form

Field Sizes

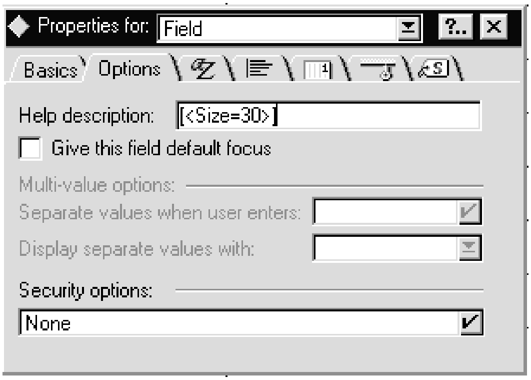
Each of the fields on the form is given a default fixed width by adding some simple HTML code into the Help Description field on the Options tab for the Field Properties InfoBox.

- To set a default width of 30, specify:

  `[<Size=30>]`

- To set a maximum number of 15 characters that can be typed into a field, specify:

  `[<Size=30 Maxlength=15>]`



Properties for: Field

Help description: [<Size=30>]

□ Give this field default focus

Multi-value options:

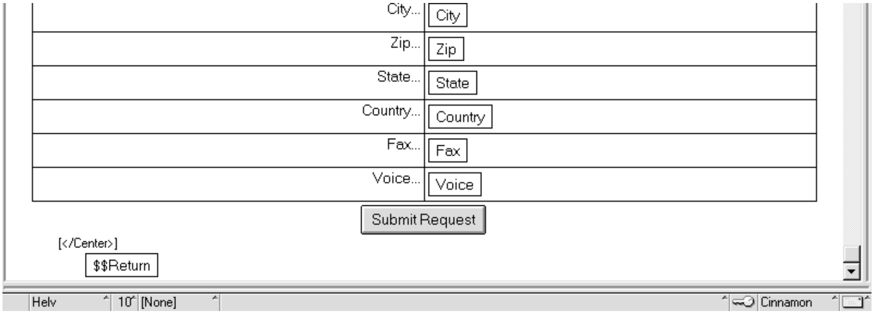
Separate values when user enters:

Display separate values with:

Security options:

None

Towards the bottom of the form is the submit button that is used to save the form from a Web client to Notes. If you do not provide a button on the form the Notes HTTP server will generate a default one for you labeled *Submit*. The advantage of adding a button yourself is that you can control the text that is displayed. The button has **NO** formula behind it.



Finally, there is a field named *$$Return*. This field can be used by you to control what the Notes HTTP server does after the form is submitted. By default, when saving a new form, the message displayed back to the user is **Form Processed** which is, perhaps, not the most informative way of telling the user that their form has been sent. In our example we want to link the user to another document that has a more informative message. This is the formula in the $$Return field,

```
"[http://cinnamon.lotus.com/Millcour.nsf/d2711fa5d4717f028525
639400489e1c/85789efc7e9b028c8525639400487d17?OpenDocument]"
```

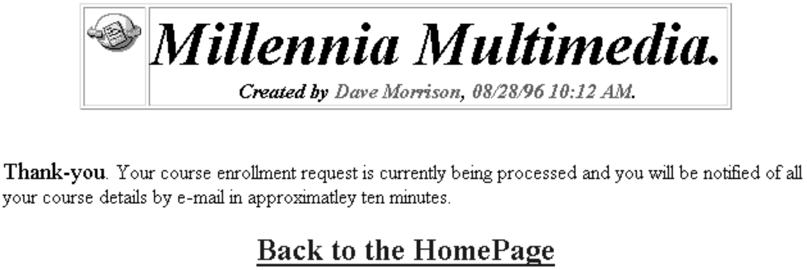
At first glance it looks too complex to understand but it is simply telling the Notes HTTP server to open a document in a view. The first of the two long numbers denote to Notes a view name and the second is a document ID within that view which is opened with the ?OpenDocument command.

Tip The easiest way to code this link to a document is to copy the full URL from a Web browser. First open the view that the document is in with a command such as *http://cinnamon.lotus.com/Millcour.nsf/TempView?Open* which will open a view named TempView that contains your document and opens it. Depending on your Web client, you should be able to copy the full URL of the document onto the clipboard and paste it into your $$Return field.

The reason that we link to an existing form in the database and not simply compose a new one, is that when you use the compose function to create a new document the Notes HTTP server will automatically add a Submit button to the end of the form which would confuse the user.

Thank-You Form

Here is the form that is displayed to the user once they have clicked the Submit Request button.



Millennia Multimedia.

Created by Dave Morrison, 08/28/96 10:12 AM.

Thank-you. Your course enrollment request is currently being processed and you will be notified of all your course details by e-mail in approximatley ten minutes.

Back to the HomePage

This is a very simple form that displays a text message to the user and displays a link back to the home page. This link is achieved by adding the following text directly onto the form,

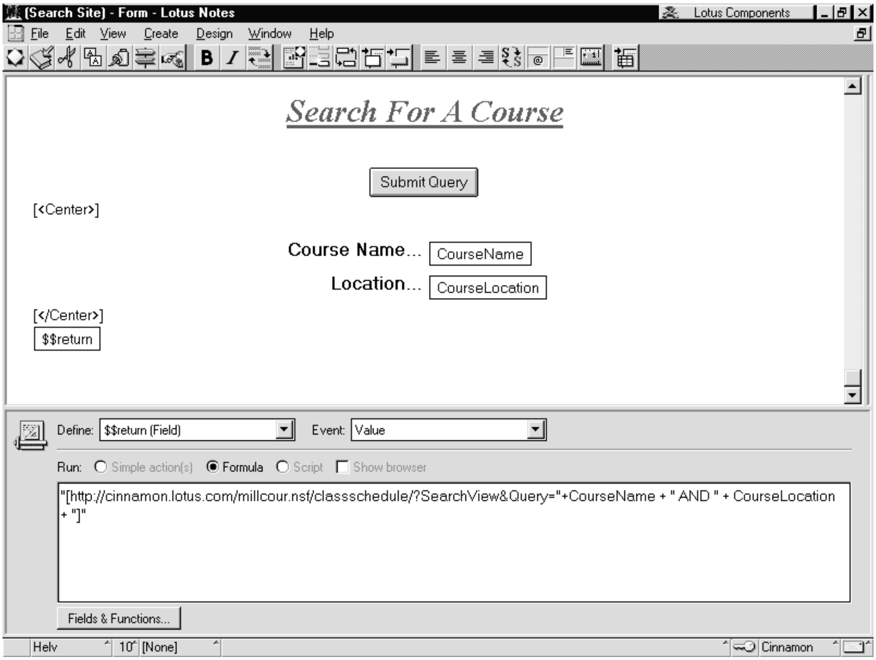
```
[<A HREF="http://cinnamon.lotus.com/Millcour.nsf/Home
Page?OpenNavigator"> <H2>Back to the Home Page</H2></A>].
```

This is an anchor to a Web page using HTML that is sent directly to the Notes HTML server. This command says, open the Home Page navigator in the Millcour.nsf file when the text Back to the Home Page is clicked.

Search Site Form

Having now created a workflow where a Web user can visit the site, browse the information and register for a course if they wish, it would be advantageous if the user could search for a particular course being held at a certain location.

This is what the search form is used for. The following is a diagram of the search form when displayed in design mode.



Search For A Course

[<Center>]

Course Name... CourseName

Location... CourseLocation

[</Center>]

$$return

Define: $$return (Field) Event: Value

Run: ○ Simple action(s) ● Formula ○ Script □ Show browser

```
"[http://cinnamon.lotus.com/millcour.nsf/classschedule/?SearchView&Query="+CourseName + " AND " + CourseLocation
+ "]"
```

The form contains two input fields for users to select the course they are searching for and the location of the course. A $$return field is used to actually process the search query and a button to initiate the search.

The first field, CourseName is a keywords field that has the following formula,

```
@Unique(@DbColumn(""; ""; "Courses"; 1))
```

This returns a list of all courses currently available from a view named Courses.

Similarly, the second field, CourseLocation is a keywords field and has the following formula,

```
@DbColumn(""; ""; "Location"; 1)
```

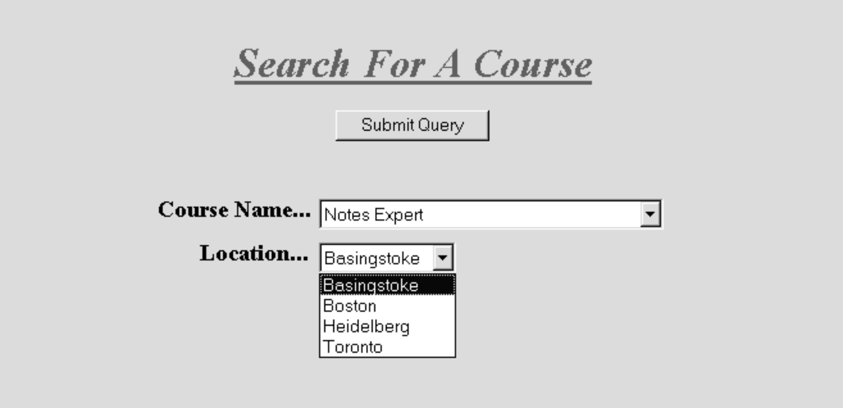
The *$$return* field has the following formula,

```
"[http://cinnamon.lotus.com/millcour.nsf/classschedule/?Searc
hView&Query="+CourseName + " AND " + CourseLocation + "]"
```

which actually submits an HTML request containing the query to Notes.

- Millcour.nsf is the name of the Notes Database
- ClassSchedule is the name of a Notes view within the database
- ?SearchView is the Notes HTTP server command to initiate a search on the view
- &Query= is the text that you are searching for within the view. In our example, if we search for the course name Notes Expert and the location is Basingstoke then the query command becomes, &Query=Notes Expert AND Basingstoke.

This is how the form looks when displayed using a Web client.



Search For A Course

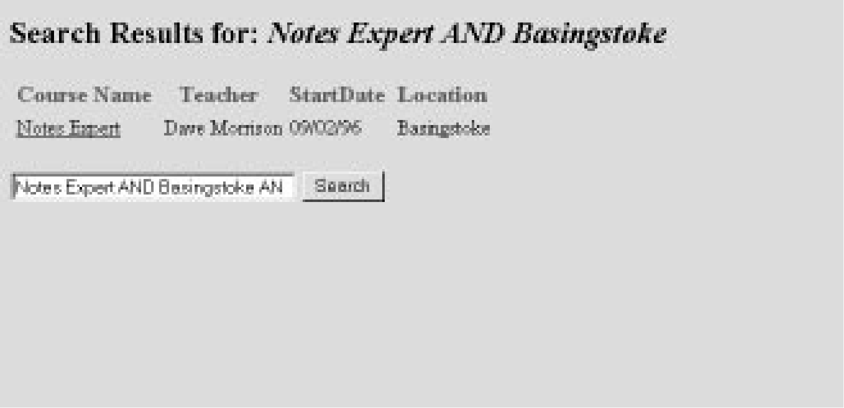
Course Name... Notes Expert

Location... Basingstoke

Note Keyword lists become drop-down list boxes when viewed in a Web client.

The HTML string that is sent to the Notes HTTP Server for the above query is, http://cinnamon.lotus.com/millcour.nsf/classschedule/?SearchView&Query=Notes+Expert+AND+Basingstoke. The HTTP server automatically connects each word in the search phrase with a + symbol.

This is how the resulting search looks after it has been run.



Search Results for: ***Notes Expert AND Basingstoke***

Course Name	Teacher	StartDate	Location
Notes Expert	Dave Morrison	09/02/96	Basingstoke

The user can then click on the course name of one of the returned documents and open it.

Agents

E-mail Reply Agent

After the Web user has registered for a course, an agent is triggered on the server to send them back an e-mail message, through the SMTP gateway, to say that they have successfully been registered.

Here is a simplified version of the agent's code.

```
Sub Initialize
     Dim db As NotesDatabase
     Dim Message As String
     Set sess = New NotesSession

     Set db = sess.CurrentDatabase
     Set docs = db.UnProcessedDocuments

     For d = 1 To docs.count
          '** Get the next document in the list
          Set doc = docs.getNthDocument(d)
          '** Create a new document to be used the mail
          'document
          Set maildoc = New NotesDocument(db)

          '** Set the Form type to a memo
          maildoc.form = "Memo"
          '** Set the person to send this to as the email
          'address adding the SMTP gateway to the address
          maildoc.SendTo = doc.Email & " @ SMTP"
          '** Set the document title
          maildoc.Subject = "Course Confirmation"
          '** Write the body of your text in here.
          maildoc.Body = "Your course reservation has " & _
          "been confirmed"
          maildoc.send(False)
          '** Mark the document as processed (so we don't
          'mail them again)
          sess.UpdateProcessedDoc(doc)
     Next
End Sub
```

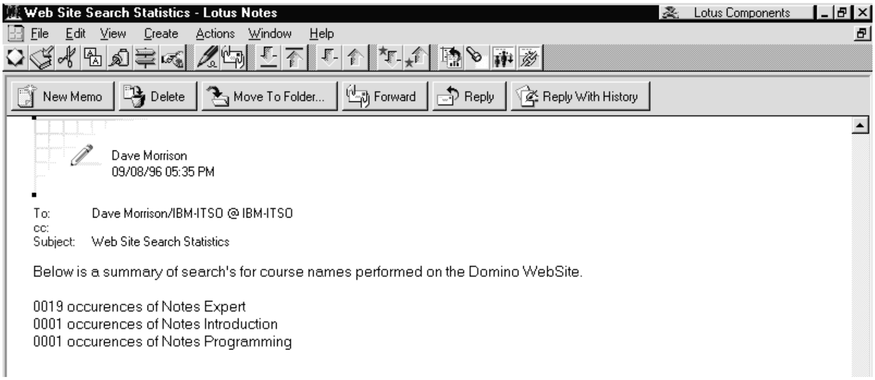
Web-Site Search Summary Agent

When a user visits the Web site and performs a search for a course, the search request document is stored in Notes. We can use this information to find out what visitors to the site are looking for and build a picture of the most common requests.

The agent was triggered to run once a week and send a Notes mail message to a designated person. Here is the code we used to create this agent.

```
Option Public
'** Work from base 1 and not 0.
Option Base 1

'** Create our own datatype called searchrec
Type SearchRec
     SearchString As String
     Count As Integer
End Type

Sub Initialize
     Dim db As NotesDatabase
    '** Create an undimensioned array from our SearchRec
     Dim Rec() As SearchRec
     Dim Found As Integer
     Dim rt As NotesRichTextItem

    '** Get the current notes session
     Set sess = New NotesSession
    '** Get the current database
     Set db = sess.CurrentDatabase
    '** Set docs to be an array of docs not processed before
     Set docs = db.UnProcessedDocuments

    '** Dimension the rec array to 1
     Redim rec(1)
    '** Set doc to be the first doc found
     Set doc = docs.getNthDocument(1)
    '** Fill the first array element with data
    '** This tells us that so far we have found one
    '** occurrence of the first text string
     rec(1).SearchString = doc.CourseName(0)
     rec(1).Count = 1

     found = False

    '** Now loop through the remaining documents
     For d = 2 To docs.count
         '** Get the next document
          Set doc = docs.getNthDocument(d)
          '** Check to see if we already have an occurrence
          '** of this search request
          For i = 1 To Ubound(rec)
               If doc.CourseName(0) = rec(i).SearchString
Then
                   '** If we do then increment count by 1
                    rec(i).Count = rec(i).Count + 1
                    found = True
               End If
          Next
         '** If we have not found a copy of this search yet
         '** then it must be a new one.
          If found = False Then
               '** Resize the array, keeping the old contents
               Redim Preserve rec(Ubound(rec) + 1)
               '** Populate our new array element
               rec(Ubound(rec)).SearchString = _
                   doc.CourseName(0)
               '** and set the count to 1
               rec(Ubound(rec)).Count = 1
          End If
          found = False
         '** Now delete this document from the database
          doc.remove(True)
     Next

    '** Having now gone through all the documents we have
    '** an array of text strings and a count on how many
    '** times they occured. We now can send a document with
    '** these details to a designated person.

    '** Create a new document
     Set doc = db.CreateDocument
    '** Set the form type to a memo document
     doc.form = "Memo"
    '** Set the person it is to be sent to.
     doc.SendTo = "Dave Morrison/CAM/Lotus"
    '** Set the title for the document
     doc.Subject = "Web Site Search Statistics"
    '** Create a link to the Body field.
     Set rt = New NotesRichTextItem(doc, "Body")
    '** Write some text to the body field.
     rt.AppendText("Below is a summary of search's for " & _
     "course names performed on the Domino WebSite.")
    '** Add two blank lines
     rt.AddNewLine(2)

    '** Loop through our array and add a line for each search
    '** word found and how often it was searched for
     For i = 1 To Ubound(rec)
          rt.AppendText(Format(rec(i).Count, "0000") & _
          " occurences of " & rec(i).SearchString)
          rt.AddNewLine(1)
     Next
    '** Send the document.
     doc.Send(True)
End Sub
```

Below is how the memo looks when it has been sent to the designated person. An alternative to this memo would be to embed a Lotus Chart Component into the body field and populate the chart with the data from the array. For an explanation of Lotus Components, see the chapter on Lotus Components in this book.



Dave Morrison
09/08/96 05:35 PM

To: Dave Morrison/IBM-ITSO @ IBM-ITSO
cc:
Subject: Web Site Search Statistics

Below is a summary of search's for course names performed on the Domino WebSite.

0019 occurences of Notes Expert
0001 occurences of Notes Introduction
0001 occurences of Notes Programming

Chapter 16
Accessing Relational Database Management Systems With Notes

This chapter describes tools and techniques that can be used to access data resources from a Notes application.

The following areas are covered.

- LS:DO (LotusScript:Data Object)
- @DBCommand, @DBLookup, @DBColumn using ODBC
- Oracle LSX (LotusScript Extension)
- ODBC in Lotus Spreadsheet Component

All of the above features except for Oracle LSX are based on the Open Database Connectivity (ODBC) technology.

When you are developing your Notes application, you need to decide what your data access needs are and which products best meet those needs. Each product has different functionality as well as performance and programmability. When you have completed this chapter, you should know when to use which tool, and how to use it.

Data Resource Access

When you develop a Notes application, you often need to implement data integration between Notes and other data resources such as RDBMS, spreadsheet data, and ASCII delimited text files. In enterprise Notes application development, this becomes even more vital as you will surely have to integrate legacy database resources in your design.

About the Database Access Facilities

The following tools enable Notes applications to connect to data resources through ODBC or native database access.

1. LS:DO (LotusScript:Data Object)

   This is a LotusScript Extension (LSX) which provides additional LotusScript classes for accessing other data resources via ODBC.

2. @DBCommand, @DBLookup, @DBColumn using ODBC

   These are @functions for ODBC data access. The functions @DBLookup and @DBColumn are frequently used to access Notes databases as well as ODBC-compliant databases.

3. ODBC access through the Lotus Spreadsheet Component

   This feature is used by the Lotus Spreadsheet Component.

4. Oracle LSX (LotusScript Extension)

   This is a set of LotusScript classes which provide native access to Oracle RDBMS without using ODBC.

There may be both ODBC and native tools for one back-end database such as an Oracle database. Which one you use depends on your choice according to functionality and performance of the particular tool.

The following table shows the characteristic differences between the tools:

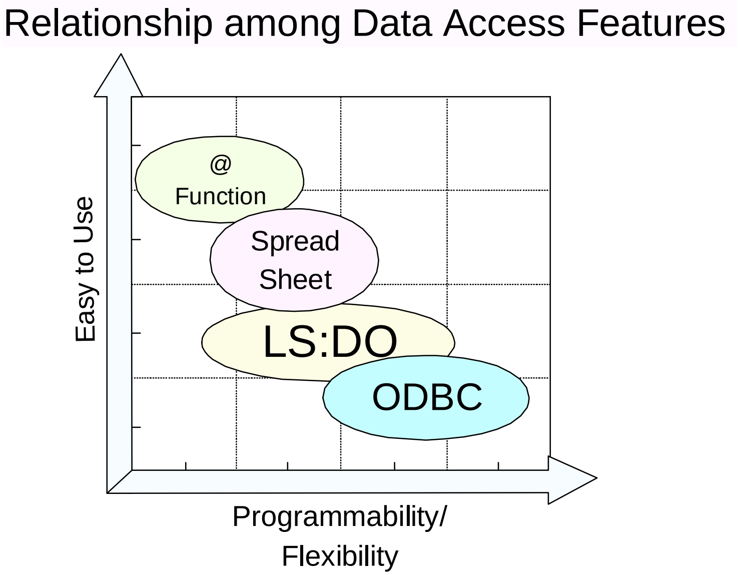
	LS:DO	*@DBLookup* *@DBColumn*	*@DBCommand*	*Spreadsheet Component*	*Oracle LSX*
Based on ODBC	X	X	X	X	
Available in LotusScript	X	*2	*2	X	X
Has a Class	X			X	X
Read Only		X	X	*1	
OCX				X	
64KB Data Limit		X	X		

Note *1 SQL statement can be specified, but the "Select" statement is usually used.
*2 Technically, LotusScript can perform @Functions under the Evaluate function.

Furthermore, we must consider some ease of use versus programming functionality and flexibility. For example, @Functions are useful to retrieve small data on the fly without complex sequences, but they are limited in the number of ways to access data.

LS:DO can be used in more complex situations with much more flexibility from a programming perspective, for example, result set handling to read and update records queried by SQL. LS:DO is also easy to use if you are familiar with LotusScript.

Study the following picture to decide which tool is most appropriate.


Relationship among Data Access Features
Easy to Use
@ Function
Spread Sheet
LS:DO
ODBC
Programmability/ Flexibility

LotusScript:DataObject (LS:DO)

What Is LS:DO?

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: LotusScript.

The LS:DO consists of a set of three classes: ODBCConnection, ODBCQuery, ODBCResultSet. These classes come complete with a powerful set of properties and methods and full SQL capabilities. Yet at the same time, the LS:DO is easy to learn and use because its design is consistent with LotusScript's BASIC syntax and other LotusScript Notes classes.

Concepts

The LS:DO is available on both the Notes client and the Notes server. LS:DO is excellent for real-time data access from any LotusScript event in Notes, such as clicking a button, exiting a field, or opening a document. LS:DO real-time data access is the best choice for the following:

- Optimizing data entry
  - On-the-fly lookups
  - Immediate updates
  - Input validation
  - Avoiding duplicate entries
- Mobile user queries and updates

Optimizing Data Entry

Many designers use Notes as the data entry point for an application, which may synchronize that data with a DBMS or use the DBMS for long-term data storage and archiving. The LotusScript Data Object can provide the following functionality on the fly:

- **On-the-fly look up**

  Once a user enters a customer name and exits the field with the TAB key or a mouse click, LotusScript code can immediately perform an SQL query to one or several external back ends, retrieve the customer record matching that name, and fill the remaining fields in the form, such as address, city, phone, and contact name.

- **Immediate updates**

  LotusScript gives you the flexibility to update the information in the relational DBMS the moment the user saves a new document in Notes or in batches at scheduled intervals. When another document is created in Notes, you can be sure that document will access the most current information in the relational DBMS.

- **Input validation**

  Is the right salesperson assigned to that customer in the Notes form? Is the regular salesperson for that region currently overloaded with assignments, indicating that a backup person should be assigned to the task? The LotusScript Data Object can retrieve that information from the DBMS that indicates these conditions, and LotusScript's fully structured programming constructs enable you to evaluate that data and act accordingly.

- **Avoiding duplicate entries**

  Once a user enters a customer's name, the LS:DO can query the back end for variations on that customer name, for example, to ensure that the same customer is not entered with an "Inc." as opposed to a "Co." in the DBMS.

Mobile User Queries and Updates

One of the most exciting results of the intersection between DBMSs and Notes is that the mobile Notes user can take their access to the DBMS with them on the road. For example, when a sales representative is on the road, they often find themselves with last-minute opportunities to visit customers in different cities. If they are on the road and if that customer information is contained in the mainframe DBMS, they are forced to call someone in the office, and ask them to look up the information, which is out of the question from a hotel room outside business hours.

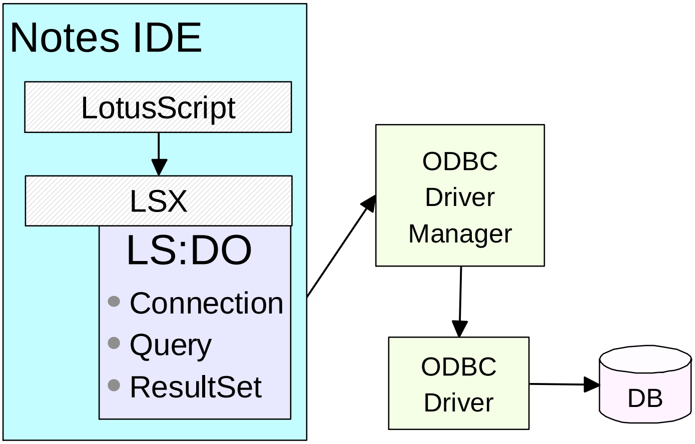
The LotusScript Data Object's ability to run on Notes servers as well as clients, coupled with Notes native replication capabilities, solves the problem. With an integrated Notes / DBMS application, a user can do the following:

1. Compose a query request within an application on their mobile Notes client, such as "What are the customer contacts and activity in this city?"
2. Replicate the query to the Notes server, where a waiting LS:DO agent sees the new document, authenticates and performs the query, stores the results in that document, and saves it.
3. Replicate the query results back to their laptop in moments, even during the same dial-up connection if they choose, for analysis and review.

Architecture

In addition to allowing users to issue SQL statements to relational DBMSs, the LS:DO also offers data manipulation capabilities. The LS:DO supports and manages result sets as well as provides an interface for directly using SQL when appropriate. The result set management takes the form of caching result sets, supporting navigation through the result set, and managing individual row updates regardless of the underlying driver's cursor or ODBC conformance capabilities.

The following diagram is a schematic representation of the components in the LS:DO framework that allow a Notes application to access a database:



When to Use LS:DO

The LS:DO is best suited to handle the following situations:

- LotusScript programming environment

  If you develop an application with the LotusScript environment, you can easily utilize ODBC access through LS:DO classes.

- Low-volume data transfer

  LS:DO is more suited for low volume access to data resources. From a performance perspective LS:DO is not well suited to moving large volumes of data.

- Easy data access

  When your application needs to both read and update data in an RDBMS, LS:DO is an easier way than the ODBC API or the @DBCommand because of the classes allowing you to work with result sets.

- Real-time direct access

  LS:DO is integrated directly in a Notes application and so on.

What Is ODBC?

The ODBC (Open Database Connectivity) standard is a set of functions established by Microsoft to access Relational Database Management Systems like Oracle, DB/2, Informix and others. There are two software components required to use ODBC:

1. ODBC Driver Manager

   This manager is a set of APIs in the ODBC dynamic link library. Those APIs are called by client programs like LS:DO, NotesSQL, and so on, in order to access an RDBMS via ODBC.

2. RDBMS ODBC driver

   The specific ODBC driver for an RDBMS. For example:

   - Oracle ODBC driver
   - NotesSQL ODBC driver
   - Desktop Database Driver

   The ODBC driver allows you to issue any SQL statements in DDL (Data Definition Language), DCL (Data Control Language) and DML (Data Manipulation Language) using SQLExecute or SQLExecDirect with the ODBC API.

   In addition, other ODBC Drivers enable you to get information about columns attributes, index, privileges of column, drivers, foreign keys of tables, and other RDBMS entities.

Using ODBC Connections

There are two ways to use ODBC:

1. You can use ODBC APIs in your C, C++, Basic, LotusScript, or any other programming language programs.

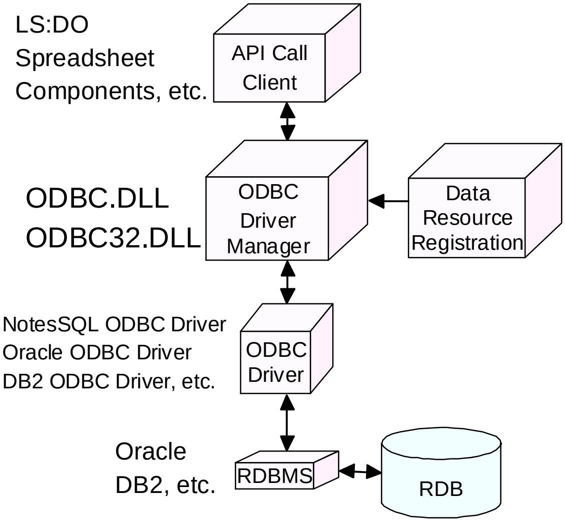
   Note The programming language you use must support calls to a DLL (Dynamic Link Library) as all of the ODBC functions are in the ODBC DLL.

2. Use ODBC compliant high level tools such as LS:DO, Lotus Spreadsheet Component in LotusScript, and Data Access Object in Visual Basic.

ODBC Access Flow

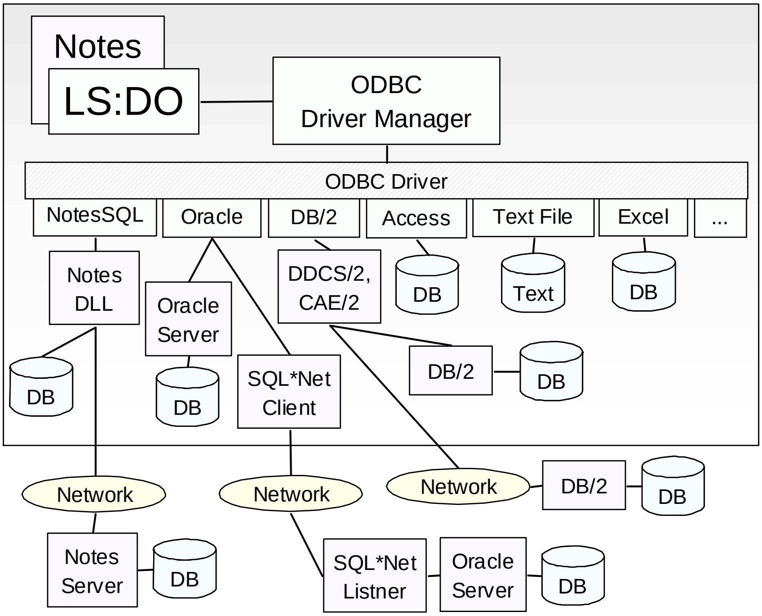
The process by which a program accesses a database through ODBC is shown below:

1. The program makes a call to the ODBC API.
2. The ODBC driver manager parses the requested command.
3. The ODBC driver manager decides which ODBC driver is required according to database resource information registered in advance through the operating system.
4. The requested command is passed to the specific ODBC driver for the database being accessed.
5. The ODBC driver composes a series of commands for the particular RDBMS and sends them to the RDBMS.
6. The results, if available, are sent back to the calling routine.



There are many ODBC drivers. Usually they are provided by RDBMS vendors but others come from independent software vendors like InterSolv or Visigenic.

The following figure conceptually shows how LS:DO makes connection paths to RDBMSs as an example. It also shows other ODBC drivers which are capable of accessing ASCII delimited Text files, Spreadsheets and other types of data resources rather than RDBMS.



Note There are two types of ODBC driver managers in the Windows environment: the 16-bit ODBC driver manager and the 32-bit ODBC driver manager. You must ensure that the one you use matches the application evironment you are in. For example, when you use the 32-bit Windows version of Lotus Notes, you need a 32-bit ODBC driver manager and a 32-bit RDBMS driver.

Difference Between LS:DO and ODBC

LS:DO is a high-level abstraction of the ODBC feature, which enables you to make more complicated operations toward RDBMS, but requires more detailed knowledge about the ODBC architecture and ODBC APIs. Let's look at three aspects of both methods:

- Programming Environment
- Functionality
- Performance

Programming Environment

Calling ODBC APIs requires passing many arguments. You have to be careful with the different argument types. A wrong argument type may cause unexpected severe errors and may make your system unstable.

The LS:DO is more intuitive and at a higher level of abstraction. Also, the LotusScript development environment checks syntax on the fly.

LS:DO is available only in LotusScript and some development environments which are compliant with OLE clients, such as Visual Basic.

LS:DO is one of the LotusScript Class Libraries (LSXs). This enables you to benefit from the object-oriented and event-driven programming environment provided by the Notes Integrated Development Environment.

Functionality

Through LS:DO classes, you can update data in a result set, which is then automatically reflected to the original table. It is much easier to update data using LS:DO methods than using an SQL statement.

ODBC functions are calls from C or C++ programs to the Dynamic Link Library. There are three conformance levels: Core Level, Extension Level 1, and Extension Level 2. There are more than 50 functions depending on the version number of the ODBC driver manager and the ODBC driver.

The following table conceptually shows which method in LS:DO calls which ODBC APIs. Each LS:DO method corresponds to a combination of some ODBC APIs:

Functions	*LS:DO Method*	*ODBC API*
List all data sources registered	Connection.**ListDataSources**	SQLAllocEnv SQLDataSources SQLFreeEnv
Establish a connection to DB	Connection.**ConnectTo**	SQLAllocConnect SQLBrowseConnect SQLFreeConnect SQLAllocConnect SQLConnect
List all tables in a database	Connection.**ListTables**	SQLAllocStmt SQLTables SQLFetch SQLGetData
Execute an SQL	ResultSet.**Execute**	SQLSetStmtOption SQLExecDirect
Fetch data from a result set	ResultSet.**GetValue**	SQLNumResultCols SQLColAttributes SQLFetch SQLGetData

Performance

ODBC provides better performance than LS:DO in some cases because the C++ program can directly access the ODBC driver manager, whereas LS:DO has some overhead due to the language architecture. Although LS:DO can make it easy to retrieve and update records in a result set, ODBC API calls are more powerful allowing the use of more complex and efficient record data handling using fetch and retrieve of records, parameterized SQL, and cursor features.

Software Requirements

The software requirements are:

- ODBC driver manager 2.0 or later.

  You need to install the appropriate ODBC driver manager as required by your operating system and by the applications that use the ODBC features.

  The 32bit ODBC driver manager comes with Visual Basic 4.0, Lotus SmartSuite 97, Office95, Visual C++ 4.0, and others.

  The 16bit ODBC driver manager comes with Windows 3.1 and Windows 95.

- ODBC drivers for specific RDBMSs.

  For example, if you create a program to access the Oracle DB server, you must install an Oracle ODBC driver which corresponds to the ODBC driver manager type already installed.

  There are many drivers provided by many software companies for RDBMSs and other data resources as well.

  Some of them are listed in the following table.

	Microsoft	*Intersolv*	*Visigenic*	*Lotus*
Notes				X
1-2-3				X
Access	X			
DB2		X	X	
dBASE	X			X*1
Excel	X			
FoxPro	X			
Informix		X	X	
Ingres		X	X	

continued

	Microsoft	*Intersolv*	*Visigenic*	*Lotus*
SQLServer	X			
Oracle	X	X	X	X*1
Paradox	X			
Sybase		X	X	
Text File	X	X		X*1

Note *1 This driver is bundled in Lotus Smartsuite.

How to Register ODBC Data Sources

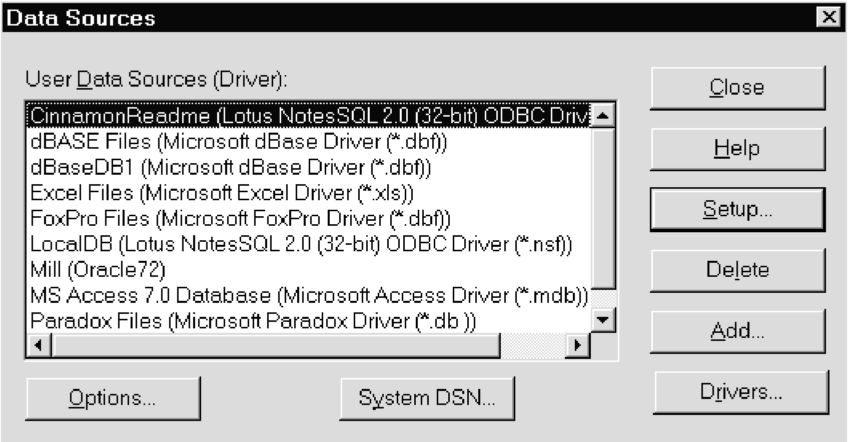
To register ODBC data sources, follow these steps. Our example is based on an Oracle DB Server connection for Windows 95. The basic operations are practically the same on the other platforms.

1. Double-click the ODBC driver manager icon in the Control Panel.

   32-bit
   Administrator

   Note The above icon image and icon title depend on the driver manager you installed on your system.

2. The Data Sources dialog box displays. You can see all data sources previously defined. To add a new data source, click the Add... button.

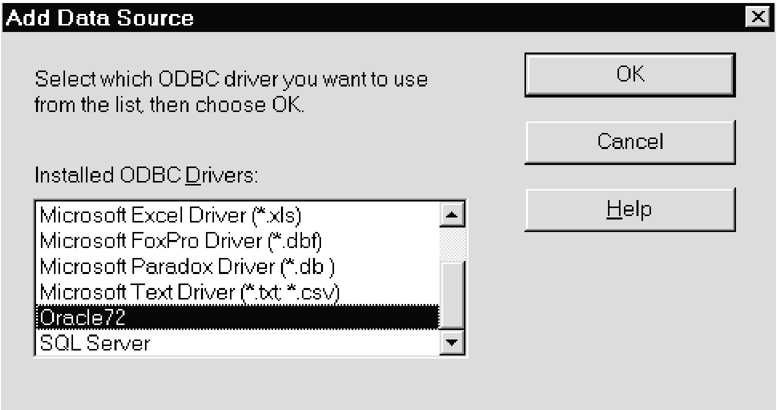


Data Sources

User Data Sources (Driver):

CinnamonReadme (Lotus NotesSQL 2.0 (32-bit) ODBC Driv
dBASE Files (Microsoft dBase Driver (*.dbf))
dBaseDB1 (Microsoft dBase Driver (*.dbf))
Excel Files (Microsoft Excel Driver (*.xls))
FoxPro Files (Microsoft FoxPro Driver (*.dbf))
LocalDB (Lotus NotesSQL 2.0 (32-bit) ODBC Driver (*.nsf))
Mill (Oracle72)
MS Access 7.0 Database (Microsoft Access Driver (*.mdb))
Paradox Files (Microsoft Paradox Driver (*.db))

3. The Add Data Source dialog box displays. You can see all the ODBC drivers installed on your PC. Select the ODBC driver for your application and click OK. In our example, we selected the Oracle 72 ODBC driver:



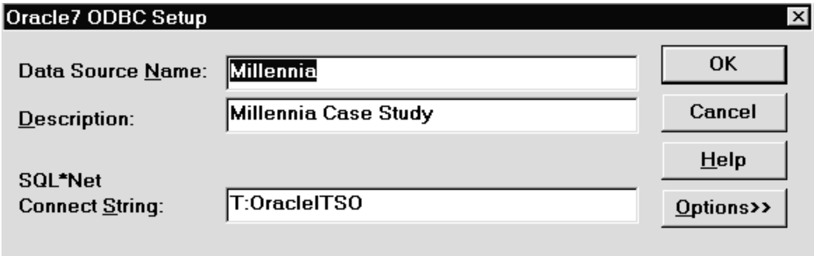
Add Data Source

Select which ODBC driver you want to use from the list, then choose OK.

Installed ODBC Drivers:

- Microsoft Excel Driver (*.xls)
- Microsoft FoxPro Driver (*.dbf)
- Microsoft Paradox Driver (*.db)
- Microsoft Text Driver (*.txt; *.csv)
- Oracle72
- SQL Server

4. If you select the Oracle ODBC driver, you'll need to fill in the appropriate information as follows. This dialog box will vary depending on the driver you choose.



Oracle7 ODBC Setup

Data Source Name: Millennia

Description: Millennia Case Study

SQL*Net Connect String: T:OracleITSO

- The Data Source Name you enter will be the one you specify in your programs, such as the ConnectTo method in LS:DO, whenever you connect to the Oracle Database.
- The Description field is just an explanation for this data source.
- The SQL*Net Connect String is important to establish the connection. In our case, the "T" means that the TCP/IP protocol is used during communication. The rest of the string "OracleITSO" is a TCP/IP hostname for the Oracle DB server on Windows NT. Optionally, you can specify the Oracle instance ID here. For a full description and different connect string examples, click on the Help button of the Oracle7 ODBC Setup dialog box.

USELSX Statement to Enable LS:DO

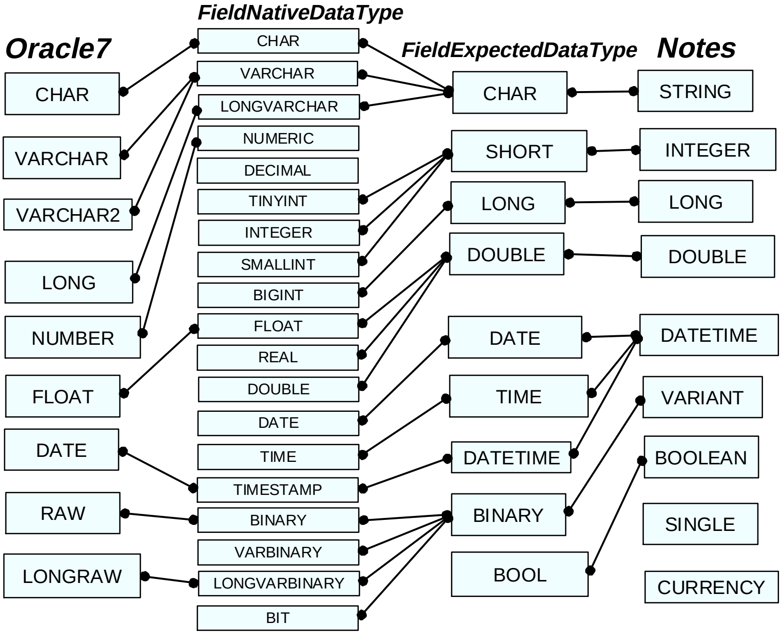
The following statement must be specified in the Define (Globals) Event (Declarations) within Lotus Notes.

```
Uselsx "*LSXODBC"
```

Note The leading "*" tells LotusScript to use the class registry to look up the path of the LS:DO dynamic library being loaded. This is a platform-independent way of loading LS:DO since each operating system uses different methods.

Mapping Data Types Between RDB and Notes DB

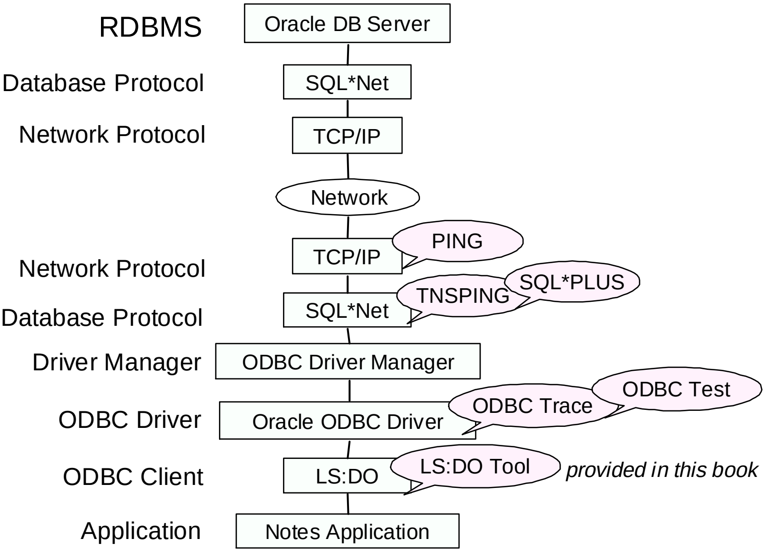
The following diagram shows the data type mapping between an Oracle database and a Notes database through the LS:DO:



How to Trace and Debug LS:DO

In this section, we will briefly touch on a few ways to debug and trace applications that employ the LS:DO.

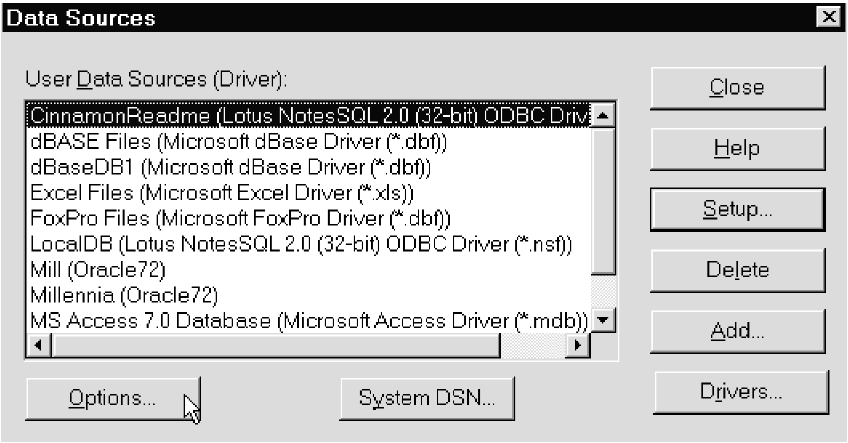
The structure for a connection between a Notes application using the LS:DO and the target RDBMS is the same as in all ODBC-compliant systems. The following diagram shows each connection layer and the respective component. If a connection cannot be established for some reason, the configuration of each of the components must be verified one by one. The appropriate debugging or tracing tool is indicated by the caption circle for each component.



Using the ODBC Trace Option

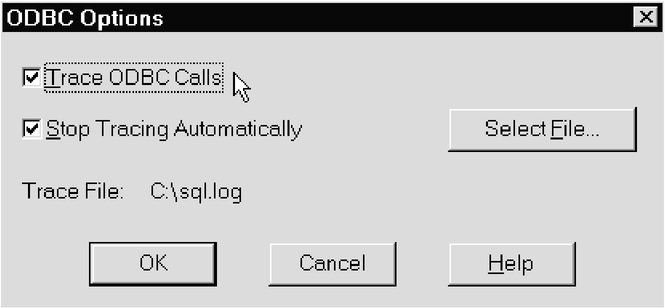
Trace information issued by the ODBC API can be gathered using the trace function in the ODBC Administrator program. To do this, follow these steps:

1. Run the ODBC Administrator program and click Options....



CinnamonReadme (Lotus NotesSQL 2.0 (32-bit) ODBC Driv
dBASE Files (Microsoft dBase Driver (*.dbf))
dBaseDB1 (Microsoft dBase Driver (*.dbf))
Excel Files (Microsoft Excel Driver (*.xls))
FoxPro Files (Microsoft FoxPro Driver (*.dbf))
LocalDB (Lotus NotesSQL 2.0 (32-bit) ODBC Driver (*.nsf))
Mill (Oracle72)
Millennia (Oracle72)
MS Access 7.0 Database (Microsoft Access Driver (*.mdb))

2. Select the Trace ODBC Calls checkbox. Click OK, and leave the ODBC Administrator program.



ODBC Options

☑ Trace ODBC Calls

☑ Stop Tracing Automatically

Trace File: C:\sql.log

Trace descriptions of the ODBC API calls in your program(s) using LS:DO are saved in the “C:\sql.log” text file as a default. You may change the log file name by clicking the Select File button in the dialog box above. An example of some of the trace output is shown below:

```
SQLAllocConnect(henv004993F0, phdbc00483E0C);
SQLConnect(hdbc00483E0C, “dBaseDB1”, -3, “”, -3, “”, -3);
SQLGetInfo(hdbc00483E0C, 11, rgbInfoValue, 4, pcbInfoValue);
SQLGetInfo(hdbc00483E0C, 21, rgbInfoValue, 4, pcbInfoValue);
SQLAllocStmt(hdbc00483E0C, phstmt00488110);
```

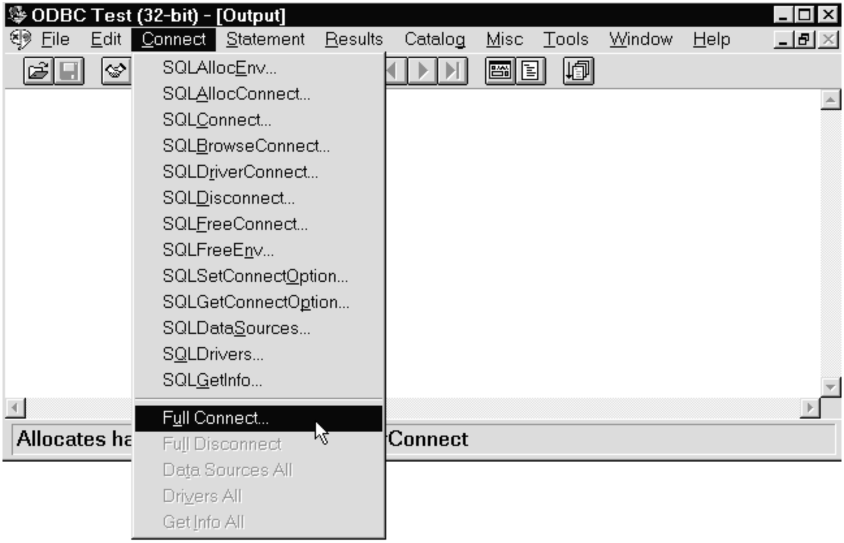
Using the ODBC Test Tool

If you are using the Microsoft ODBC driver SDK, you can use the ODBC Test tool which allows you to issue ODBC API function calls. In the following example, we will test whether the connection to the database is configured properly and, if so, try other calls to debug each of the layers.

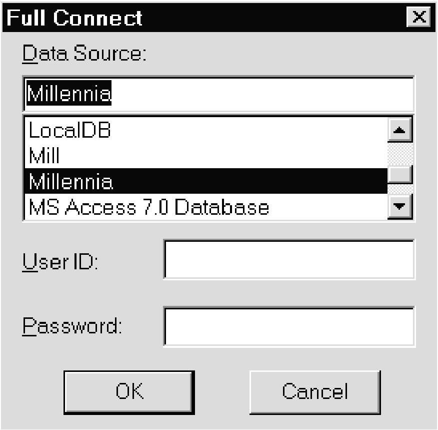
1. Double-click the following icon to start the ODBC Test program.

Odbcte32.exe

2. Choose Connect - Full Connect from the menu bar.

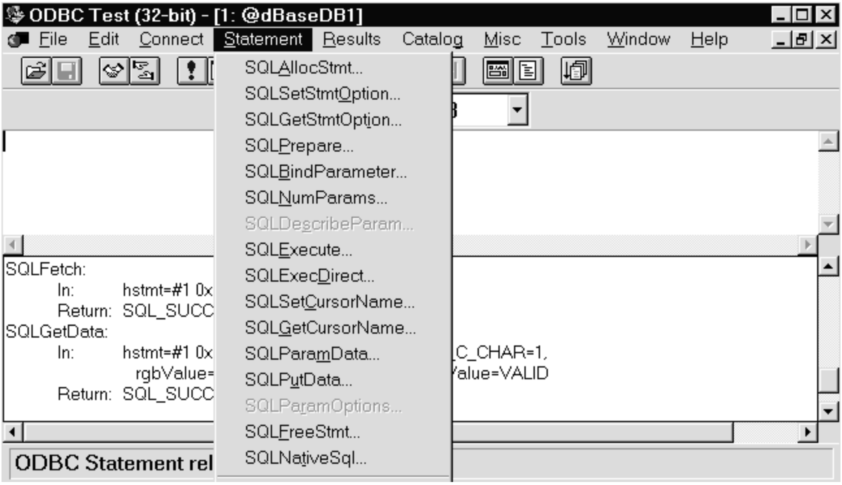


3. You are prompted for the connection specifics. From the list box, choose the Data Source you want to test the connection to. These are the data sources you specified with the ODBC driver manager. In our example, we chose the Millennia Oracle7 instance data source. If security is enforced, you must specify the User ID and / or a password.


Full Connect

Data Source: Millennia

4. You will see the results of the connection. If successful, you can select from the other ODBC API function calls from the menu to test the functionality of the connection and the results of the calls:



```
SQLFetch:
        In:       hstmt=#1 0x
        Return:   SQL_SUCC
SQLGetData:
        In:       hstmt=#1 0x                                C_CHAR=1,
                    rgbValue=                               /alue=VALID
        Return:   SQL_SUCC
```

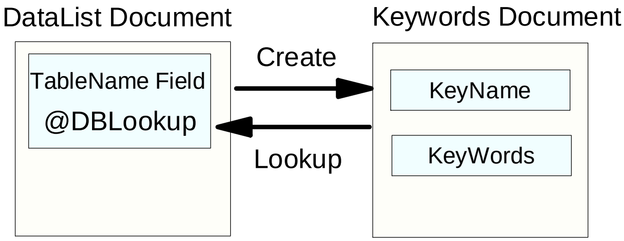
Creating Your Own LS:DO Test Application

To test the functionality of your LS:DO configuration you can build your own test application.

The goal of the tool is to provide a graphical interface which allows you to select a database name, a table name and column names which you want to check the values for. The tool returns the records in the table to a field in a form which can be stored and displayed in a view.

This tool allows you to take a deeper look at LS:DO function calls and behavior. It is meant to give you a starting point from which you can debug your LS:DO code and environment. You may want to make it more robust by adding more functionality like error handling routines, appropriate error messages, and consistency checking routines.

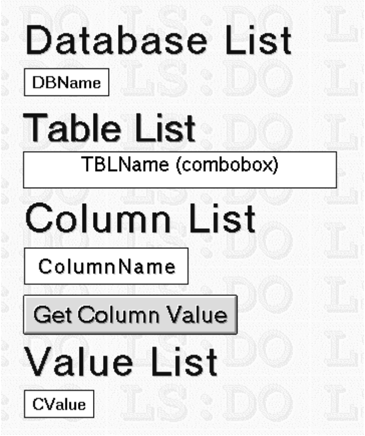
The following figure shows the flow of data in the application which dynamically populates a keyword list using the Keywords form. A keyword list document is created from a DataList document. The created keywords are retrieved using the @DBLookup function.



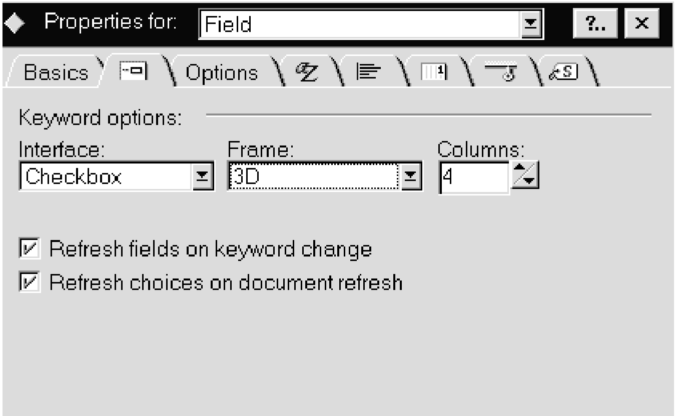
Form Design of the LS:DO Tool

The following figure shows the form design of our LS:DO tool. The application is based on a Keywords form. It has four fields to indicate the lists of selections and the results. Here are the field definitions:

- DBName is a keyword list field in which a database name can be selected and displayed as an option button list.
- TBLName is a keyword list field in which a table name can be selected and displayed as a combo box which is located in a layout region.
- ColumnName is a keyword list field in which column names can be selected and displayed as a check box list. Any column names you want can be chosen.
- CValue is a text field in which all the records composed of all the columns selected are displayed. Be aware that the size of a text field is limited to a maximum of 64KB.



The field properties are set using the following InfoBox to provide a multi-column 3D check box interface. Also, make sure you select the two check boxes provided on the Keyword options tab of the InfoBox.



Properties for: Field

Keyword options:

Interface: Checkbox

Frame: 3D

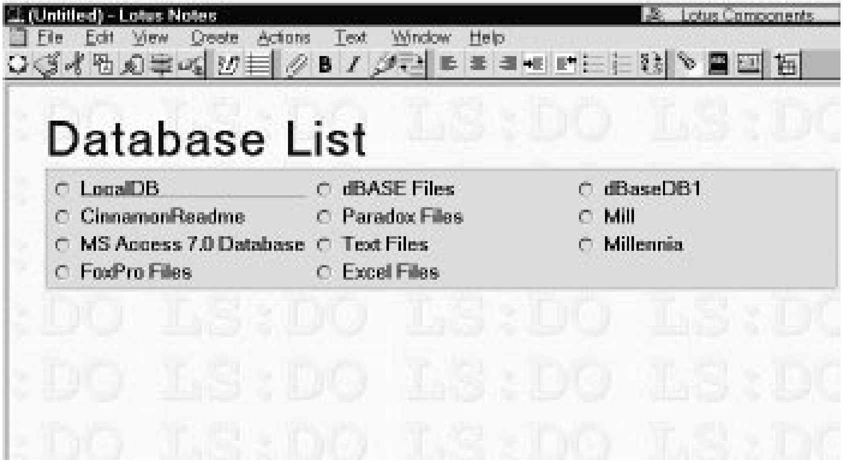
Columns: 4

☑ Refresh fields on keyword change

☑ Refresh choices on document refresh

The following steps show you how to use this tool:

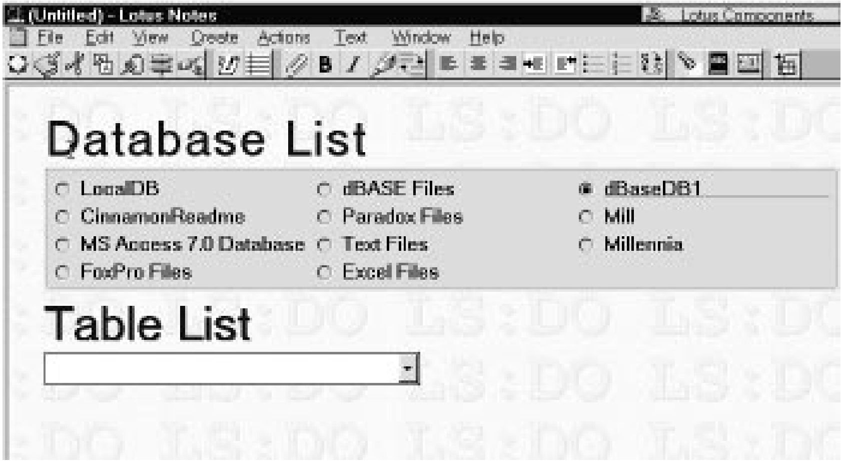
1. Create a document with the DataList form.You will see a database name list with option buttons to choose from the list:



Database List

○ LocalDB	○ dBASE Files	○ dBaseDB1
○ CinnamonReadme	○ Paradox Files	○ Mill
○ MS Access 7.0 Database	○ Text Files	○ Millennia
○ FoxPro Files	○ Excel Files	

2. Choose dBaseDB1 as a sample database. A dialog box displays that also contains a Table List combo box:

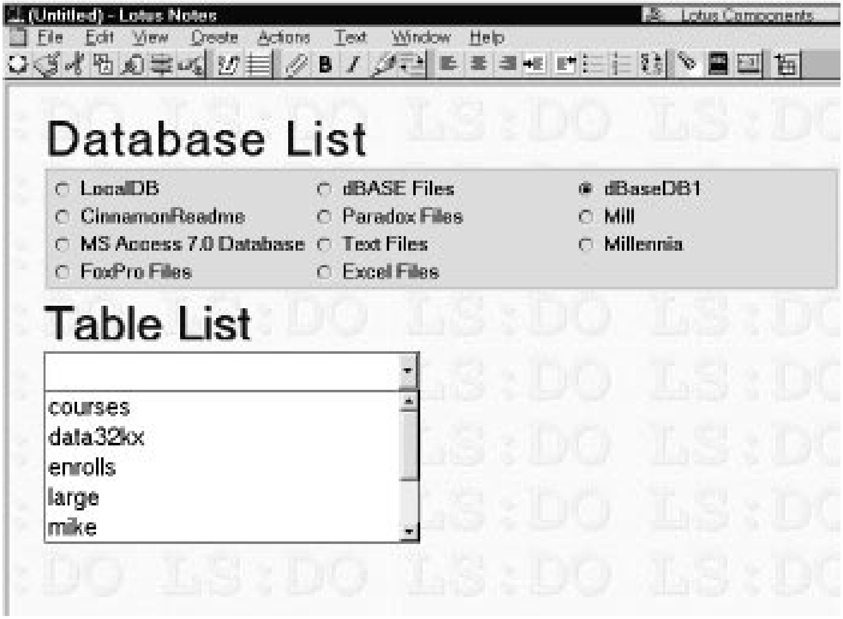


Database List

○ LocalDB ○ dBASE Files ● dBaseDB1
○ CinnamonReadme ○ Paradox Files ○ Mill
○ MS Access 7.0 Database ○ Text Files ○ Millennia
○ FoxPro Files ○ Excel Files

Table List

3. Drop down the list of available options. A list of database table names is displayed:



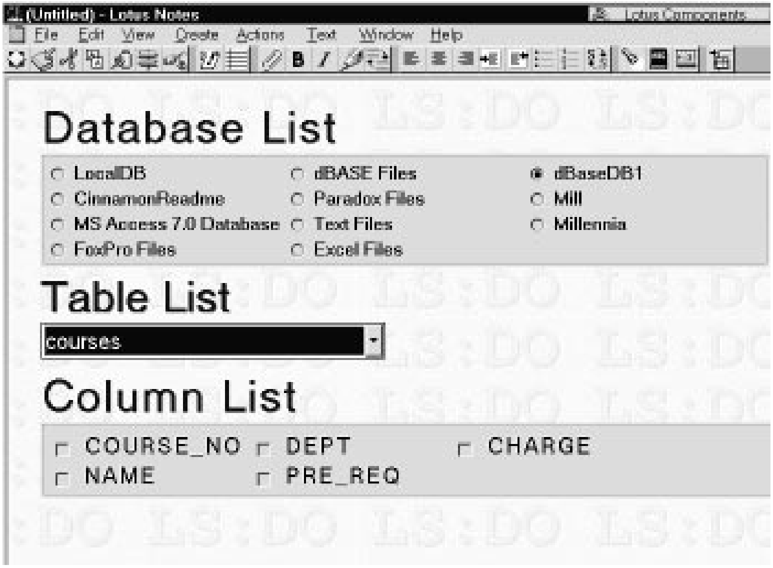
Database List

○ LocalDB ○ dBASE Files ● dBaseDB1
○ CinnamonReadme ○ Paradox Files ○ Mill
○ MS Access 7.0 Database ○ Text Files ○ Millennia
○ FoxPro Files ○ Excel Files

Table List

courses
data32kx
enrolls
large
mike

4. The Exiting event of the Table List keyword field is a trigger to populate the Column List which is presented next as a check box list:



Database List

○ LocalDB ○ dBASE Files ● dBaseDB1
○ CinnamonReadme ○ Paradox Files ○ Mill
○ MS Access 7.0 Database ○ Text Files ○ Millennia
○ FoxPro Files ○ Excel Files

Table List

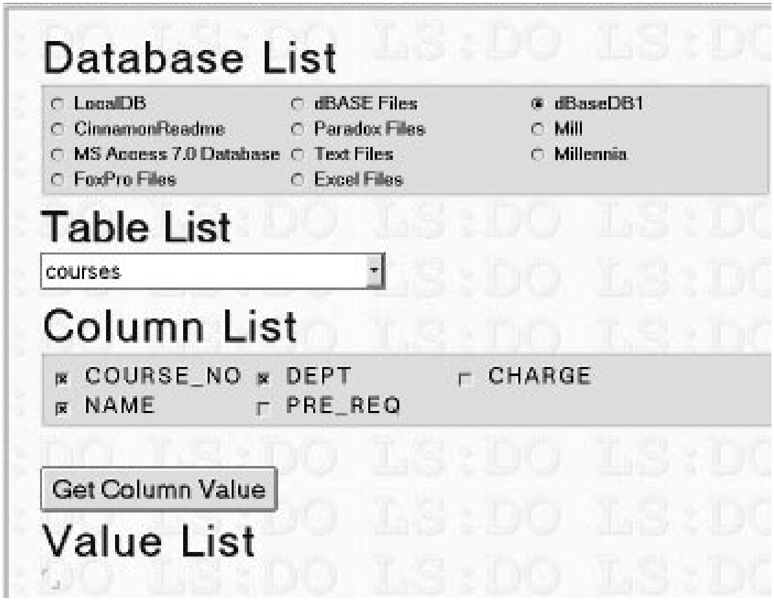
courses

Column List

□ COURSE_NO □ DEPT □ CHARGE
□ NAME □ PRE_REQ

5. Select some column names which you want to retrieve. In our example, we are accessing the Millennia course table, and we would like to retrieve the columns COURSE_NO, DEPT, NAME. The Get Column Value button is displayed right after selecting one of the column names.

6. Click the Get Column Value button to retrieve the columns:



Database List

○ LocalDB ○ dBASE Files ● dBaseDB1
○ CinnamonReadme ○ Paradox Files ○ Mill
○ MS Access 7.0 Database ○ Text Files ○ Millennia
○ FoxPro Files ○ Excel Files

Table List

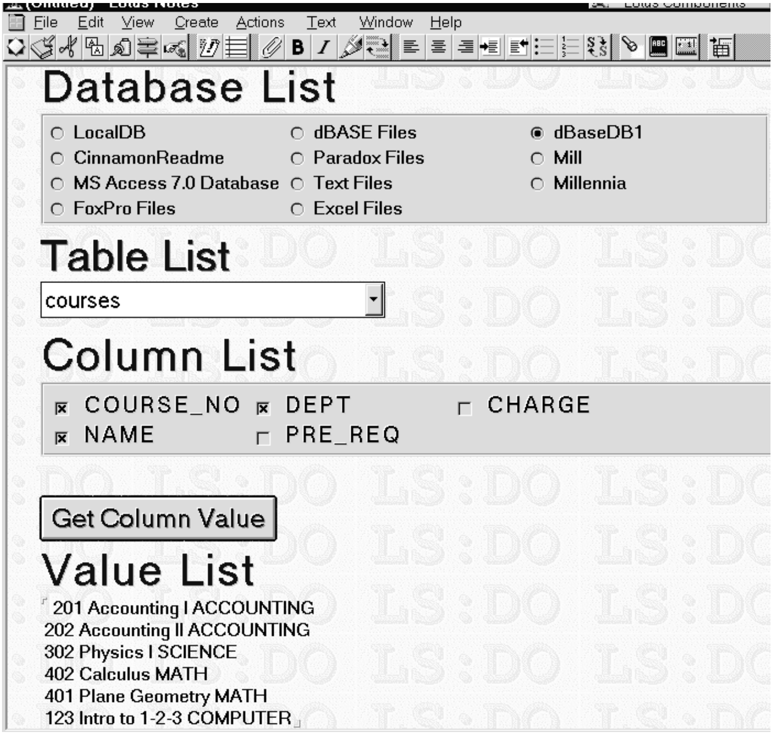
courses

Column List

☒ COURSE_NO ☒ DEPT □ CHARGE
☒ NAME □ PRE_REQ

Value List

7. You will see the query results in the Value List field. In our example, six records were retrieved:



Database List

○ LocalDB ○ dBASE Files ● dBaseDB1
○ CinnamonReadme ○ Paradox Files ○ Mill
○ MS Access 7.0 Database ○ Text Files ○ Millennia
○ FoxPro Files ○ Excel Files

Table List

courses

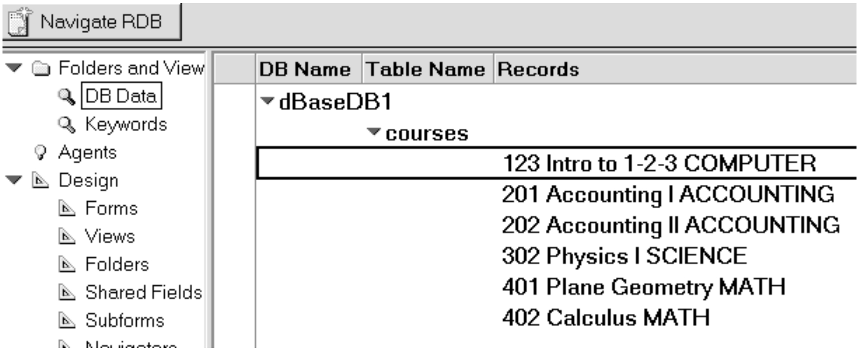
Column List

☒ COURSE_NO ☒ DEPT □ CHARGE
☒ NAME □ PRE_REQ

Value List

```
201 Accounting I ACCOUNTING
202 Accounting II ACCOUNTING
302 Physics I SCIENCE
402 Calculus MATH
401 Plane Geometry MATH
123 Intro to 1-2-3 COMPUTER
```

8. Save the result of your query into a document. You can check it again in the DB Data view. The document is displayed as multiple lines according to the number of records that were retrieved.



DB Name	Table Name	Records
dBaseDB1		
	courses	
		123 Intro to 1-2-3 COMPUTER
		201 Accounting I ACCOUNTING
		202 Accounting II ACCOUNTING
		302 Physics I SCIENCE
		401 Plane Geometry MATH
		402 Calculus MATH

LS:DO Class Library

The classes that make up the LotusScript:Data Object provide you with the following benefits:

- **Connection sharing**

  Connections are cached to avoid the added overhead of establishing a connection. In addition, since it is defined as an independent object, one connection object can be used by multiple LotusScript SQL calls.

- **Multiple query and result sets**

  You can define multiple query objects to generate multiple result set objects which can all be executed against the same connection, and manipulated from the same script.

- **Bi-directional scrolling over result set**

  The ODBCResultSet object provides a scrolling cursor with methods for navigating to the next, previous, first and last rows.

- **Result set search**

  The LocateRow method of the ODBCResultSet object provides the ability to search for specific rows within the result set based on specified criteria. This search capability executes faster than multiple queries or comparing values from multiple rows in LotusScript.

- **Cached results**

  The query result in the ODBCResultSet object is optionally cached in memory (default setting), so it can be later accessed by other events in the form, increasing performance and reducing DBMS connection time. In addition, the cached result set gives you the ability to later locate records using LocateRow.

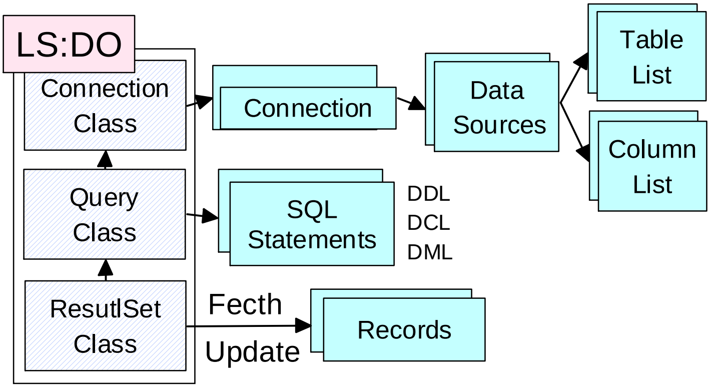
- **Update services**

  Updates to back-end DBMSs through a generic ODBC interface are limited to SQL statements, where the user must ensure that the row to be updated contains a unique record reference or can be otherwise uniquely accessed through a cursor. LS:DO extends this capability by permitting individual items in a result set to be modified without use of an SQL statement using the SetValue method. These changes are then updated to the back end database all at once.

- **Driver transparency**

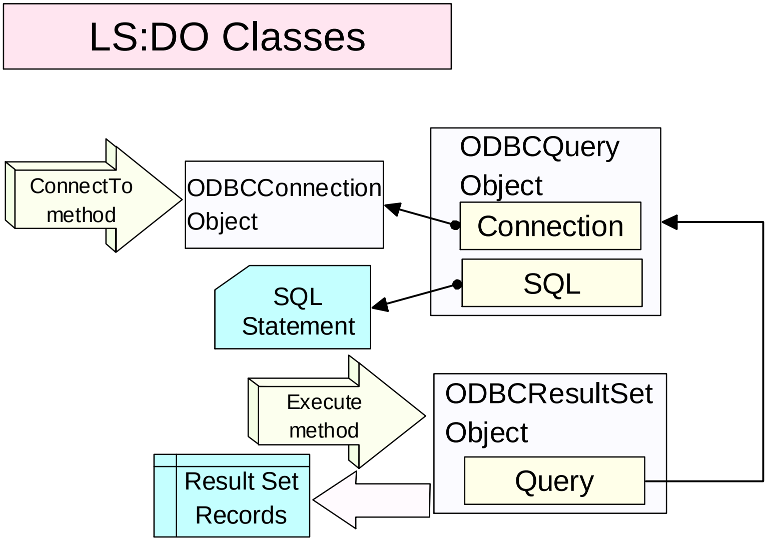
  Although different vendors' ODBC drivers support varying conformance levels, the LotusScript:Data Object assesses these differences and often provides the same level of behavior across all drivers and databases. The developer does not have to write separate scripts for separate drivers.

The following figure represents the manner in which a LotusScript program would use each class in an application access in a database.



Relationship Among Classes

The three classes in the LS:DO are tightly related to one another as shown in the following diagram.



Event Handling

If needed, you can create event handling subroutines for some ODBC methods. An event handling subroutine you create is called according to the behavior of an appropriate ODBC method, after the On Event statement is issued.

In the following example, an event handler, named presub1 is called, before the ListDataSources method is called.

1. On Event statement

```
Dim connection As New ODBCConnection
On Event BeforeListDataSources From connection Call
presub1
```

2. Event handler

```
Sub presub1(Source As ODBCConnection)
     '** Write your event handling script here
End Sub
```

Note Your event handler must be in the scope where the event occurs.

ODBCConnection Class

The ODBCConnection class allows you to establish a connection. It also allows you to access some database catalog information, such as data source lists, table lists, procedures lists and so on.

Property

The following table shows the properties of the ODBCConnection Class:

Property	*Data Type*	*Read/ Write*	*Argument*
DataSourceName	String	R	
DisconnectTimeOut	Integer	R/W	
Exclusive	Boolean	R/W	
IsConnected	Boolean	R	
IsSupported(option)	Boolean	R	option: DB_SUPP_ASYNCHRONOUS DB_SUPP_CURSORS DB_SUPP_PROCEDURES DB_SUPP_READONLY DB_SUPP_SILENTMODE DB_SUPP_TRANSACTIONS
IsTimedOut	Boolean	R	
SilentMode	Boolean	R/W	

Note Boolean is not a pre-defined data type in LotusScript. But you can use a constant value (TRUE and FALSE) as a Boolean data type.

Method

The following table shows the ODBCConnection methods with the corresponding arguments and events.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
ConnectTo(source$ [, userID$, password$])		Boolean	DBstsCANF DBstsSVRQ DBstsCCON DBstsACCS	BeforeConnect AfterConnect BeforeConnectTo AfterConnectTo
Disconnect		Boolean	DBstsNCON	BeforeDisconnect AfterDisconnect TransactionUpdate
ExecProcedure(name$, arg$)		Boolean	DBstsNCON DBstsODBC	BeforeExecProcedure AfterExecProcedure
GetError		Constant *1		
GetErrorMessage([error%])	error%: DB_LASTERROR or Constants *1	String		
GetExtendedErrorMessage([error%])	error%: DB_LASTERROR or Constants *1	String		
GetRegistrationInfo(source$)		String	DBstsCANF	
ListDataSources		Array of String		BeforeListDataSources AfterListDataSources
ListFields([tableName$])		Array of String	DBstsNCON DBstsNCOL	BeforeListFields AfterListFields
ListProcedures([source$ [, userID$, password$])		Array of String	DBstsNCON DBstsACCS	BeforeListProcedures AfterListProcedures
ListTables([source$ [, userID$, password$]])		Array of String	DBstsNCON DBstsACCS	BeforeListTables AfterListTables

*1 Error number list is shown in ODBCResultSet section.

Note Most of the pre-defined data types in LotusScript are represented by the following suffix types:

Data Type	*Suffix*
Currency	@
Double	#
Integer	%
Long	&
Single	!
String	$

Sample Uses of the ODBCConnection Class:

1. To get a data source list registered by the ODBC administrator, use the ListDataSources method of the ODBCConnection class.

```
Dim con As New ODBCConnection
Dim dl As Variant
dl = con.ListDataSources
'** Keywords is a field name in which a data source list
is saved
keyDoc.Keywords = dl
```

2. To get a table list owned by a database, use the ListTables method of the ODBCConnection class.

```
Dim con As New ODBCConnection
Dim tl As Variant
'** sampleDB1 is a database name registered in this
example
tl = con.ListTables("sampleDB1")
'** Keywords is a field name in which a table list is
saved
keyDoc.Keywords = tl
```

   Note When the ListTables method is issued, the SQLConnect ODBC API is called in LS:DO before getting a table list. So you don't need to execute the ConnectTo method.

3. To get a column name list owned by a table, use the ListFields method of the ODBCConnection class.

```
Dim con As New ODBCConnection
Dim Clist As Variant
Dim status As Variant
'** sampleDB1 is a database name registered in this
example
status = con.ConnectTo("sampleDB1")
'** courses is a table name in the sampleDB1 database
CList = con.ListFields("courses")
'** Keywords is a field name in which a column list is
saved
keyDoc.Keywords = Clist
```

ODBCQuery Class

The ODBCQuery class is used to hold the ODBCConnection object in which a connection is established, and to hold an SQL statement you want to use to perform the inquiry. The SQL statement is parsed through the ODBC driver which your application requires.

Property

The three properties of this class are shown below:

Property	*Data Type*	*Read/Write*
Connection	ODBCConnection Object	W
QueryExecuteTimeOut	Integer	R/W
SQL	String	R/W

Method

The ODBCQuery class provides the following methods:

Method	*Argument*	*Return Value*
GetError		Constant
GetErrorMessage([error%])	error%: DB_LASTERROR or Constants *1	String
GetExtendedErrorMessage ([error%])	error%: DB_LASTERROR or Constants *1	String

*1 Error number list is shown in ODBCResultSet section.

Sample Uses of the ODBCQuery Class:

This sample shows a sample execution of an SQL statement using the Execute method in ODBCResultSet. The following steps are needed before executing the Execute method. The Connection method and the SQL property in ODBCResutlSet class are also used in this example.

```
Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
'** sampleDB1 is a database name registered in this
example
status = con.ConnectTo("sampleDB1")
Set qry.Connection = con
'** courses is a table name in the sampleDB1 database
qry.SQL = "select * from courses"
Set res.Query  = qry
```

ODBCResultSet Class

The ODBCResultSet class has many functions used to handle records which are termed result sets. A result set holds the retrieved records of an SQL query which is specified with the ODBCQuery object.

Property

The following table shows the properties available with the ODBCResultSet Class:

Property	*Data Type*	*Read/Write*
Asynchronous	Boolean	R/W
AutoCommit	Boolean	R/W
CacheLimit	Integer	R/W
CommitOnDisconnect	Boolean	R/W
CurrentRow	Integer	R/W
FetchBatchSize	Integer	R/W
HasRowChanged	Boolean	R
IsBeginOfData	Boolean	R
IsEndOfData	Boolean	R
IsResultSetAvailable	Boolean	R
MaxRows	Integer	R/W
NumColumns	Integer	R
NumRows	Integer	R
Override	Boolean	W
Query	ODBCQuery Object	W
ReadOnly	Boolean	R/W

Methods

The methods of the ODBCResultSet class can be categorized into the following areas:

- SQL execution and transaction control
- Result set row navigation and location
- Accessing column values
- Result set row modification operations
- Column attributes operations
- SQL parameter operations

The following tables show the methods based on the above categories.

SQL Execution and Transaction

These methods are used to issue an SQL statement and to commit or roll back a transaction.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
Close(option)	Option: DB_CLOSE DB_COMMIT DB_ROLLBACK	Boolean		BeforeClose AfterClose
Execute([option])	Option: DB_CANCEL	Boolean	DBstsODBC	BeforeExecute AfterExecute AsynchOperationComplete
Transactions(option)	Option: DB_COMMIT DB_ROLLBACK	Boolean		BeforeTransactions AfterTransactions

Result Set Row Locating Operations

These methods are used to locate a cursor on a result set which is produced by the Execute method.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
FirstRow		Boolean	DBstsINVR	BeforeFirstRow AfterFirstRow BeforeRowPositionChange AfterRowPositionChange
LastRow		Boolean		BeforeLastRow AfterLastRow BeforeRowPositionChange AfterRowPositionChange
LocateRow(column, value$ [, column, value$, ...])	column is Integer or String.	Boolean	DBstsCARR DBstsEOFD DBstsNODA	BeforeLocateRow AfterLocateRow BeforeRowPositionChange AfterRowPositionChange
NextRow		Boolean	DBstsINVR DBstsEOFD	BeforeLocateRow AfterLocateRow BeforeRowPositionChange AfterRowPositionChange
PrevRow		Boolean	DBstsINVR DBstsCARR	BeforePrevRow AfterPrevRow BeforeRowPositionChange AfterRowPositionChange

Accessing Column Value Operations

These methods are used to access specific column values and to check column properties.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
GetValue(column [, variable])	column is Integer or String.	Variant	DBstsINVC DBstsNODA DBstsCNVR	BeforeGetValue AfterGetValue
IsValueAltered(column)	column is Integer or String.	Boolean	DBstsINVC	
IsValueNull(column)	column is Integer or String.	Boolean	DBstsINVC	
SetValue(column, value)	column is Integer or String	Boolean	DBstsRDON DBstsRDEL DBstsINVC DBstsCNVR DBstsNODA	AfterSetValue BeforeSetValue

Result Set Row Modification Operations

These methods enable you to dynamically add and delete rows from within the result set. Furthermore, you can retrieve the row status and you can update the altered result set in the database.

Method	*Argument*	*Return Value Type*	*Error Constant*	*Event*
AddRow		Boolean	DBstsAHVR DBstsRDON DBstsNOEX	BeforeAddRow AfterAddRow
DeleteRow(tableName$)		Boolean	DBstsINVR DBstsNUNQ DBstsRCHG DBstsRDON	BeforeDeleteRow AfterDeleteRow RowContentsChanged TransactionsPending
GetRowStatus		DB_UNCHANGED DB_ALTERED DB_UPDATED DB_DELETED DB_NEWROW	DBstsNODA	
RefreshRow		Boolean	DBstsNUNQ DBstsINVR	BeforeRefreshRow AfterRefreshRow
UpdateRow		Boolean	DBstsRDON DBstsRDEL DBstsCXIN DBstsNUNQ DBstsRCHG DBstsRUNC DBstsUPDB	BeforeUpdateRow AfterUpdateRow TransactionsPending RowContentsChanged

Column Attributes Operations

These methods allow you to access information about the column attributes.

Method	*Argument*	*Return Value Type*	*Error Constant*
FieldExpectedDataType (column [, dataType])	column is Integer or String. dataType: DB_TYPEUNDEFINED DB_CHAR DB_SHORT DB_LONG DB_DOUBLE DB_DATE DB_TIME DB_BINARY DB_BOOL DB_DATETIME	DB_TYPEUNDEFINED DB_CHAR DB_SHORT DB_LONG DB_DOUBLE DB_DATE DB_TIME DB_BINARY DB_BOOL DB_DATETIME	DBstsINVC
FieldID(columnName$)		Integer	DBstsINVC
FieldInfo(column)	column is Integer or String.	Array with elements *1	DBstsINVC
FieldNativeDataType (columnID%)		String	DBstsINVC
FieldID(column)	column is Integer or String.	Constant *2	DBstsINVC
FieldName(column)	column is Integer or String.	Integer	DBstsINVC

SQL Parameter Operations

These methods are used to define new SQL parameters and to retrieve the values of those already existing.

Method	*Argument*	*Return Value Type*	*Event*
GetParameter(parameter)	parameter is Integer or String	Variant	BeforeGetParameter AfterGetParameter
GetParameterName (parameterID%)		String	BeforeGetParameterName AfterGetParameterName
NumParameters		Integer	
SetParameter(parameter, value$)	parameter is Integer or String	Boolean	BeforeSetParameter AfterSetParameter

Error Operations

These methods are used to deal with error messages.

Method	*Argument*	*Return Value Type*
GetError		Constant *3
GetErrorMessage([error%])	error%: DB_LASTERROR or Constants *3	String
GetExtendedErrorMessage([error%])	error%: DB_LASTERROR or Constants *3	String

*1 The following table shows return value constants of the FieldInfo method.

DB_INFO_AUTOINCREMENT	DB_INFO_NULLABLE
DB_INFO_CASESENSITIVE	DB_INFO_PRECISION
DB_INFO_COLUMNID	DB_INFO_READONLY
DB_INFO_COLUMNNAME	DB_INFO_SCALE
DB_INFO_COMPUTED	DB_INFO_SEARCHABLE
DB_INFO_DISPLAYSIZE	DB_INFO_SETTABLE
DB_INFO_EXPECTED_DATATYPE	DB_INFO_SQLDATATYPE
DB_INFO_LENGTH	DB_INFO_TABLENAME
DB_INFO_MONEY	DB_INFO_UNSIGNED
DB_INFO_NATIVE_DATATYPE	

*2 The following table shows return value constants of the FieldNativeDataType method.

SQL_CHAR	SQL_FLOAT	SQL_TIMESTAMP	SQL_LONGVARBINARY
SQL_NUMERIC	SQL_REAL	SQL_VARCHAR	SQL_BIGINT
SQL_DECIMAL	SQL_DOUBLE	SQL_BINARY	SQL_TINYINT
SQL_INTEGER	SQL_DATE	SQL_VARBINARY	SQL_BIT
SQL_SMALLINT	SQL_TIME	SQL_LONGVARCHAR	

*3 The following table shows error constants.

DBstsSUCCESS	DBstsINVC	DBstsDSTY	DBstsTMPL	DBstsRDON
DBstsFAIL	DBstsNCOL	DBstsDRVN	DBstsBROW	DBstsRCHG
DBstsMEMF	DBstsBADP	DBstsFITY	DBstsCANF	DBstsRUNC
DBstsNCON	DBstsODBC	DBstsFILT	DBstsCNVR	DBstsCXIN
DBstsCCON	DBstsLIBM	DBstsINST	DBstsCNVD	DBstsAHVR
DBstsNOEX	DBstsSNFD	DBstsNODR	DBstsHSTMT	DBstsCPAR
DBstsINVR	DBstsINTR	DBstsNAUT	DBstsSQLP	DBstsNIRC
DBstsCARR	DBstsACCS	DBstsNOSV	DBstsINTE	DBstsRDEL
DBstsNODA	DBstsTYPE	DBstsNAPE	DBstsUPDB	
DBstsEOFD	DBstsENTR	DBstsSVRQ	DBstsNUNQ	

Sample Programs Using the ODBCResultSet Class

In this example we get a column value list.

```
Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
'** sampleDB1 is a database name registered in this
example
status = con.ConnectTo("sampleDB1")
Set qry.Connection = con
'** courses is a table name in the sampleDB1 database
qry.SQL = "select * from courses"
Set res.Query  = qry
Call res.execute
Dim num As Integer
num = 0
Dim vl As Variant
Redim vl(num)
Do
'** name is a column name in the courses table
     vl(num) = res.GetValue("name")
     num = num + 1
     Redim Preserve vl(num)
Loop While res.NextRow
'** CValue is a field name in which a value list is saved
gDoc.CValue = vl
```

The Millennia Multimedia Case Study: An Example Program

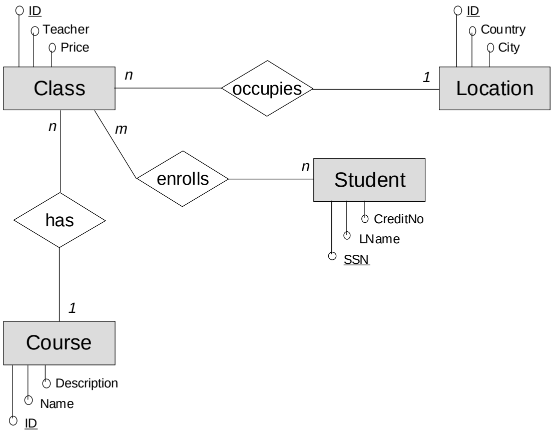
In this section, we will present a more elaborate example to demonstrate how to access a database using the LS:DO classes.

The Millennia Multimedia Database Schema

Continuing with the case study introduced in the chapter on Developing Web Applications, Millennia Multimedia uses a relational database management system (here: Oracle) to hold course and schedule information.

This database stores information about available courses and their schedules. A course schedule is given by one or more planned classes with a certain date, location, a teacher, and a maximum number of people who can participate. All students for a given course are registered with the required billing information.

The entity relationship (E-R) diagram for this database looks as follows:



Using this E-R diagram, we can define a logical database schema. The following relations need to be created. The primary keys of the relations are underlined:

Class (ID, CourseID, LocationID, Teacher, StartDate, EndDate, Limit, Language, Price, Currency, LocationID)

Course (ID, CName, CDesc, CType)

Student (SSN, FName, LName, Company, Title, HasPaid, CreditNo, EMail)

Location (ID, Country, State, ZIP, City, Street, Building)

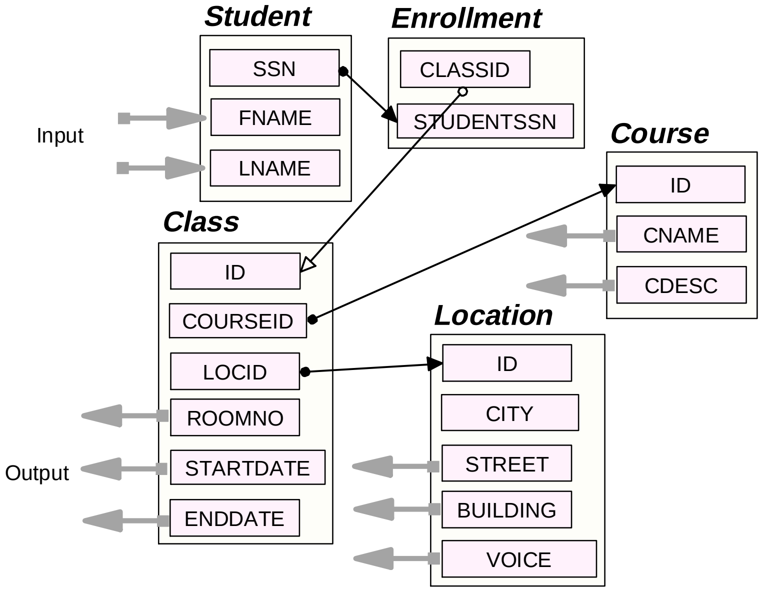
Enrollment (ClassID, StudentSSN)

To keep the relations as simple as possible, not all attributes that you would expect in a real business database are included.

Sample SQL Query

In this example we use the presented database schema which was implemented on an Oracle database server to write a script that retrieves all enrollments for a given person's name.

The following figure is a graphical representation of the query:



- The SQL statement

  The following SQL statement is based on the above relationships.

```
SELECT B.ID, B.STARTDATE, B.ENDDATE, E.CNAME, E.CDESC,
A.CITY, A.STREET, A.BUILDING, B.ROOMNO, A.VOICE
FROM LOCATION A, CLASS B, COURSE E
WHERE B.LOCID=A.ID AND B.COURSEID=E.ID AND
B.ID IN (SELECT D.CLASSID
FROM STUDENT C, ENROLLMENT D
WHERE C.FNAME='FirstName' AND C.LNAME='LastName' AND
C.SSN=D.STUDENTSSN)
```

- Sample Script

  The script is as follows:

```
Sub Click(Source As Button)
Dim ws As New NoteUIWorkspace
Dim doc as NotesDocument
Dim con As New ODBCConnection
Dim qry As New ODBCQuery
Dim res As New ODBCResultSet
Dim status As Variant
Set doc = ws.CurrentDocument.Document
status = con.ConnectTo("Millennia")
Set qry.Connection = con
'** the select statement
qry.SQL = | SELECT B.STARTDATE, B.ENDDATE, E.CNAME,
E.CDESC,A.CITY, A.STREET, A.BUILDING, B.ROOMNO, A.VOICE
FROM LOCATION A, CLASS B, COURSE E
WHERE B.LOCID=A.ID AND B.COURSEID=E.ID AND
B.ID IN (SELECT D.CLASSID
FROM STUDENT C, ENROLLMENT D
WHERE C.FNAME='| & doc.FirstName & _
"' AND C.LNAME='" & doc.LastName & _
"' AND C.SSN=D.STUDENTSSN)"
Set res.Query  = qry
Call res.execute
Dim num As Integer
num = 0
Dim vl As Variant
Redim vl(num)
Do
'** Concatenate all column values
  For colnum = 1 to 9
    vl(num) = vl(num) & " " & res.GetValue(colnum)
  Next
  num = num + 1
  Redim Preserve vl(num)
Loop While res.NextRow
'** doc.CValue is a field name in which the results are
saved.
doc.CValue = vl
Call res.Close
Call con.Disconnect
```

Using @DB Functions to Access Other Databases Through ODBC

@DBCommand, @DBLookup and @DBColumn are Notes functions that enable you to access RDBMSs which use the underlying ODBC interface. The @DB formulas are read-only.

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

When to Use

Lotus Notes provides fast and easy-to-use read access to ODBC-compliant DBMSs via @DB functions. Notes @DB functions give developers the power of three frequently-used query tasks:

- Generating Keyword Lists

  The @DBColumn function in the Notes formula language generates Notes keyword lists from internal as well as external data sources. The same function supports keyword value lookups in tables stored in a DBMS through ODBC. For example, a Notes @DBColumn field formula can present a keyword list of customer names stored in a DBMS table when composing a document in a Notes customer contact tracking database.

- Performing Lookup Operations

  The @DBLookup function looks up a value in one field based on the value of a related field. For example, it will look up a customer phone number in a DBMS when given a customer name in Notes. Like @DBColumn, @DBLookup works both with other Notes databases and with external data sources through ODBC. The @DBColumn and @DBLookup functions can be used in other Notes formula contexts as well, such as input validation or translation formulas.

- Launching External DBMS Stored Procedures

  Database procedures and insert statements can be triggered with the @DBCommand function.

How to Use @DB Functions

The @DB functions are summarized in the following table:

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

How to Use @DBColumn

The @DBColumn syntax is:

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

Parameters:

@DBColumn("ODBC":	*Description*	*Choice*	*Optional*
Cache,	Inquiry Cache	"Cache" (Default) "NoCache"	X
DataSource,	Database resouce name		
UserID1:UserID2,	User IDs		X
Password1:Password2,	Passwords		X
TableName,	Table Name		
ColumnName:	Column Name		
NULLHandling,	Null Handling	"Fail" "Discard" (Default) "ReplacementValue"	X
Distinct:	Remove duplicate values	"Distinct"	X
Sort)	Sort Direction	"Ascending" "Descending"	X

How to Use @DBLookup

The @DBLookup syntax is:

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

Parameters:

@DBLookup("ODBC":	*Description*	*Choice*	*Optional*
Cache,	Inquiry Cache	"Cache" (Default) "NoCache"	X
DataSource,	Database resouce name		
UserID1:UserID2,	User IDs		X
Password1:Password2,	Passwords		X
TableName,	Table Name		
ColumnName:	Column Name		
NULLHandling,	Null Handling	"Fail" "Discard" (Default) "ReplacementValue"	X
KeyColumn,	Column Name to be looked into		
Key,	Search String in KeyColumn		
Distinct:	Remove duplicate values	"Distinct"	X
Sort)	Sort Direction	"Ascending" "Descending"	X

How to Use @DBCommand

The @DBCommand syntax is:

```
@DBCommand( "ODBC": Cache ; DataSource  ; UserID1 : UserID2 ;
Password1 : Password2 ; SQL ;  NullHandling )
```

Parameters:

@DBCommand("ODBC":	*Description*	*Choice*	*Optional*
Cache,	Inquiry Cache	"Cache" (Default) "NoCache"	X
DataSource,	Database resouce name		
UserID1:UserID2,	User IDs		X
Password1:Password2,	Passwords		X
SQL	SQL Statement		
NULLHandling)	Null Handling	"Fail" "Discard" (Default) "ReplacementValue"	X

Using the Oracle LSX

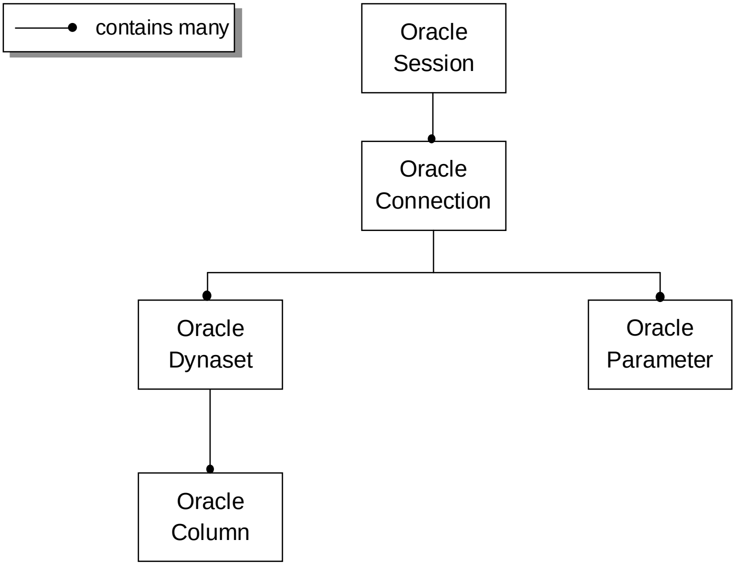
The Oracle LotusScript Extension (Oracle LSX) is a library of LotusScript classes for dedicated access to Oracle Release 7 databases. Unlike the LS:DO interface that uses the ODBC industry standard to enable access to many different database systems, the Oracle LSX object model is specific to Oracle databases. It uses the native Oracle API for its operation.

If you are planning to access only Oracle databases from within your Notes applications, you may want to use the Oracle LSX instead of LS:DO. The Oracle LSX classes offer special database access features to you, such as real SQL parameters and SQL array parameter operations to enhance the performance of Oracle database operations. Also, it avoids the additional ODBC passthru overhead for all database statements. On the other hand, if you are not sure whether or not you need to access other database systems in the future, we recommend that you rely on the much more general LS:DO object model.

In contrast to LS:DO, with the Oracle LSX you get full access to the Oracle database management system. This means that you can execute any SQL statement, including operations on table spaces, for example.

Architecture

In your Notes object event scripts, you load the Oracle LSX with a USELSX statement. From then on, you can create objects of the classes provided by the LSX to establish database connections; you can call methods on these objects to perform queries and other SQL statements. The Oracle LSX classes translate all these operations into the appropriate Oracle API calls which are then passed either to the Oracle database system on the local machine, or via SQL*Net to a remote Oracle server. The following figure illustrates this architecture:



Object Hierarchy

The Oracle LSX object model consists of five object classes which are related to each other in a containment hierarchy. These are OracleSession, OracleConnection, OracleDynaset, OracleParameter, and OracleColumn.

The containment relationship between the classes determines the way in which they are used:

1. At the beginning, you must create an OracleSession object which provides a context for all further database connections.
2. An object of the class OracleConnection is used to establish and manage a connection to a database.

3. For a given connection, you can execute arbitrary SQL statements that don't return values, and SQL queries. All statements can be parameterized using OracleParameter objects. Furthermore, you will usually create an OracleDynaset to perform operations on the query result set.
4. Each of the result set columns in the OracleDynaset are represented by objects of the class OracleColumn. They hold useful name and size information.

Using the Oracle LSX Classes

Setting Up a Session

The main access point to all Oracle database facilities is provided by an OracleSession object. The first action in your script is to create such an object. Then you can call the GetConnection method on that object to establish the first connection to an Oracle database.

Your Notes application will have exactly one session object regardless how often you create such an object. Any subsequent attempt will not return a new session object but a reference to the object that was created by the first call. This global session object maintains all connections you create. Furthermore, you create OracleParameter objects using method calls on this session object, and you can retrieve created parameter objects by name. These SQL parameters play an important role when you want to implement efficient SQL statements; we will describe them later.

Finally, the session object supports you with properties for error detection and handling.

The available methods for a session object are:

- CreateParameter and GetParameter

  SQL parameter maintenance.
- GetConnection

  Establishes a new database connection.

Establishing a Database Connection

The declaration of the GetConnection method of a session object is as follows:

```
GetConnection (connect_string,
               userid, password,
               type, commit_mode)
```

The connect_string holds the SQL*Net connect string by which you identify the database you want to connect to. The parameters userid and password are used to log you in.

The parameter type is one of the values CONNECT_SHARED and CONNECT_EXCLUSIVE. It enables you to specify whether to share an already existing connection to the same database with the same user ID and the same transaction mode (commit_mode). When you want to establish a connection to another database, or to the same database with another user ID or transaction mode, or you specify CONNECT_EXCLUSIVE, a new connection will be established.

Note Connections are a limited resource in Oracle, and establishing connection is a time-consuming operation.

The parameter commit_mode determines whether you want to commit each SQL statement immediately (COMMIT_AUTO), or by a separate method call (COMMIT_PROGRAM). The latter option is most useful when you have to deal with transactions that contain multiple SQL statements. However, you must distinguish between data definition language (DDL) statements that change the data model, and data manipulation statements (DML) that operate on data stored in tables. DDL statements are always committed immediately; the commit_mode option only affects the DML statements.

Executing DDL and Simple DML Commands

The OracleConnection class contains properties that inform you about the connection string, the user ID, and the chosen connection type and transaction mode.

You can call the following methods on a connection object:

- Commit and Rollback

  These methods either commit or undo your database operations since the last commit or rollback.

- ExecuteSQL

  You may use this method to execute any DDL and DML statement as well as any PL/SQL block. It is not intended to return any values, hence it doesn’t make sense to execute a query that returns a set of rows. Use the method CreateDynaset for that purpose.

- CreateDynaset

  This method enables you to execute queries that return a set of rows. The resulting set is returned as an OracleDynaset object through which you can navigate to perform further processing.

The following sample LotusScript code demonstrates how to use the OracleSession and OracleConnection classes. The purpose of the script is to insert 100 new courses into the Millennia database. It uses a parameterized SQL statement in conjunction with appropriate OracleParameter objects that hold the value arrays to be inserted:

```
'** Load the Oracle LSX
UseLSX "lsxorcl.dll"

'** Declaration and definition of a database session
Dim session as New OracleSession
'** Declaration of a connection object
Dim conn as OracleConnection

'** Establishing a shared connection with programmed commit
Set conn = session.GetConnection ("Millennia",
                                  "ADMIN", "PASSWORD",
                                  CONNECT_SHARED,
                                  COMMIT_PROGRAM)
'** Create SQL parameters for insertion: 5 steps
'**  (1) Create LotusScript value arrays
Dim IDValArr    (100) As Integer
Dim CNameValArr (100) As String
Dim CDescValArr (100) As String
Dim CTypeValArr (100) As Integer

'**  (2) Fill the arrays with the values to insert
...
'**  (3) Declare corrsponding OracleParameters
Dim IDOraParm    As OracleParameter
Dim CNameOraParm As OracleParameter
Dim CDescOraParm As OracleParameter
Dim CTypeOraParm As OracleParameter

'**  (4) Define the OracleParameters:
'**      Here, all parameters are input parameters.
'**      Each of them has a name, a type, and an array size.
Set IDOraParm    = session.CreateParameter ("ID",
                        PARAMETER_INPUT,
                        LS_DT_LONG_INTEGER, 100)
'**      String arrays also get the maximum string length
Set CNameOraParm = session.CreateParameter ("CName",
                        PARAMETER_INPUT,
                        LS_DT_VARIABLE_STR, 100, 30)
Set CDescOraParm = session.CreateParameter ("CDesc",
                        PARAMETER_INPUT,
                        LS_DT_VARIABLE_STR, 100, 70)
```

```
Set CTypeOraParm = session.CreateParameter ("CType",
                        PARAMETER_INPUT,
                        LS_DT_SHORT_INTEGER, 100)
'**  (5) now assign the value arrays to the parameter objects
IDOraParm.Value    = IDValArr
CNameOraParm.Value = CNameValArr
CDescOraParm.Value = CDescValArr
CTypeOraParm.Value = CTypeValArr

'** Now execute the SQL statement: it inserts 100 rows.
'** We have to use the SQL parameter names as
'** defined for the OracleParameter objects.
conn.ExecuteSQL ("Insert into Course values
                        (:ID, :CName, :CDesc, :CType)")

'** Commit the work
conn.Commit
```

Executing Queries

The declaration of the CreateDynaset method of a connection object is as follows:

```
OracleDynaset CreateDynaset (select_str
                             [,scrollmode]
                             [,pagesize] [,cachemode]
                             [,iomode])
```

The only required argument is select_str, in which you pass a select statement as a string.

The remaining arguments are intended for fine-tuning the navigation through the result set. They heavily depend on the query you specify, and on the navigation through the resulting rows:

- Oracle always fetches a fixed size of rows from the server when you navigate to a row that wasn't yet fetched. The pagesize parameter determines how many rows are fetched; the default is 100.
- The scroll mode specifies whether the dynaset maintains only one page of rows at a time, or if it attempts to hold all fetched rows.
- The cache mode indicates whether the dynaset should use main or secondary storage on the local file system as cache. The latter is the recommended mode for Windows 3.X.
- The iomode parameter specifies whether the result set is to be modified or not.

Working With Query Result Sets

The OracleDynaset returned by a CreateDynaset call represents the query result set of rows. This class maintains a cursor to a current position in the dynaset, and provides you with the following types of methods to manipulate the set:

- Navigation methods

  Using the functions GoTo (rownumber) and GoToLast, you can access any row in the result set.

- Access methods

  You can retrieve the meta information of the result columns using the method GetColumn. It returns an OracleColumn object that stores the name and the data type for the column in question.

  You access the values of individual columns of the current row just like a property of the OracleDynaset object. This means that you type

  ```
  dynaset.CName
  ```

  if your SELECT statement contains CName as result column. If you want to update the column value, simply assign the new value to that property.

  Note This great flexibility is a feature of LotusScript. Classes need not be of static type; the class implementation may extend the attribute set of a class dynamically. The OracleDynaset class implementation makes use of this mechanism, and adds the column names of the query result expression as new properties. Note that some default rules apply to computed columns. You may change any column name of the result expression by modifying the name information stored in the OracleColumn objects.

  Also, certain methods enable you to retrieve the column values of data type LONG for the current row.

- DML methods

  Using the InsertRow and DeleteRow methods, you can insert new rows into the result set, and delete existing ones.

- Database update method

  If you modify the query result set, you have to store the changes in the database by calling the SendUpdatesToDB. Otherwise the changes are lost.

The following example uses the course and schedule information stored in the Millennia database to implement a specific marketing activity. A special offer volume discount mailing is produced for all students that have attended more than three classes. The code is as follows:

```
   '** Load the Oracle LSX
   UseLSX "lsxorcl.dll"

   '** Declaration and definition of a database session
   Dim session as New OracleSession
   '** Declaration of a connection object
   Dim conn as OracleConnection

   '** Establishing a shared connection with programmed
   commit
   Set conn = session.GetConnection ("Millennia",
                                     "ADMIN", "PASSWORD",
                                     CONNECT_SHARED,
                                     COMMIT_PROGRAM)

   '** Create a dynaset related to the intended SQL query
   Dim resultSet as OracleDynaset
   Set ds = conn.CreateDynaset ( _
      " select Title, Fname, Lname, Email "  _
       & " from Student "  _
       & " where SSN in "  _
       & " ( select Studentssn "  _
       & "   from Enrollment " _
       & "   group by Studentssn " _
       & "   having count (Classid) > 3 )" )

   '** Prepare the end-of-set handling
   On Error OR_ERR_DS_EMPTY GoTo EndOfResultSet
   On Error OR_ERR_NOSUCHROW GoTo EndOfResultSet

   '** Create the mailing including all those students
   resultSet.GoTo (1)

   While True
      '** Create a memo for that student.
      '** The procedure createMemo is not shown here.
      Call createMemo (resultSet.Title, _
                       resultSet.Fname, resultSet.Lname, _
                       resultSet.Email)
      '** Move the cursor to the next row
      resultSet.GoTo (resultSet.CurrentRow + 1)
   Wend

EndOfResultSet:
   '** End of the script
```

ODBC Database Access Methods in Lotus Spreadsheet Component

The Lotus Spreadsheet component is one of the Lotus Components. It embeds the LotusScript language, and provides you with some database access functions that you can use to perform calculations based on values stored in databases.

Like the LS:DO, these access functions rely on the ODBC interface. In contrast to LS:DO, they don't include the concept of query result sets, because they are not intended to update databases but rather to put the retrieved values in sheet cells. The functions are designed to provide a very easy method to access arbitrary relational databases, and to render the results of a query. For example, the size of the spreadsheet can be dynamically changed according to a result of a query.

Note You need a platform which supports OCX to utilize Lotus Components features, such as Windows95.

How to Use ODBC Database Access Methods

There are three methods for ODBC database access.

- ODBCConnection method

  ```
  ODBCConnect( pConnect$, bShowErrors&, pRetCode% )
  ```

  pRetCode% is "Call by Reference."

- ODBCQuery method

  ```
  ODBCQuery( pQuery$, nRow&, nCol&, bForceShowDlg&,
  pSetColNames&, pSetColFormats&, pSetColWidths&,
  pSetMaxRC&, pRetCode% )
  ```

  pSetColNames&, pSetColFormats&, pSetColWidths&, pSetMaxRC& and pRetCode% are "Call by Reference."

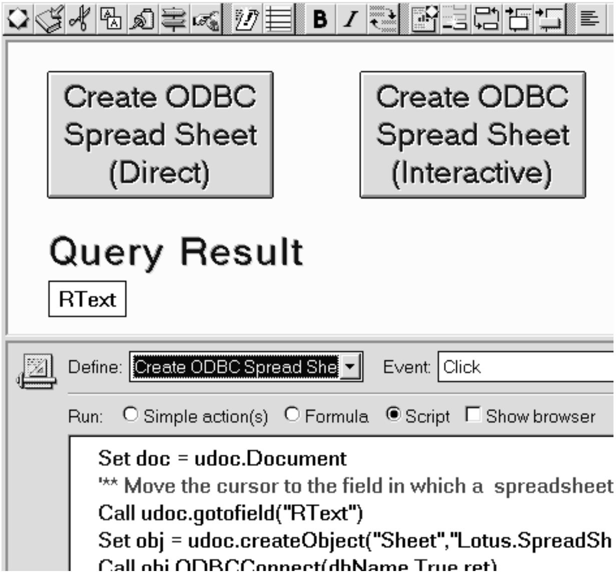
- ODBCDisconnect method

  ```
  ODBCDisconnect
  ```

The most complicated is the ODBCQuery function that supports interactive as well as non-interactive queries. While the latter query type is most useful to gain complete control over the query parameters in the script, interactive queries prompt you to specify the database, the query statement, and the result columns during runtime. Depending on the type of the query, you have to take care of the way you pass arguments. Some arguments must be passed by reference, because they return values.

Example: A Non-Interactive Query

In the following examples we demonstrate how to use the ODBC data access functions in the Lotus Spreadsheet Component. Both examples are based on a single form that contains different scripts in the two button click events.


Create ODBC Spread Sheet (Direct)
Create ODBC Spread Sheet (Interactive)
Query Result
RText

Define: Create ODBC Spread She

Event: Click

Run: ○ Simple action(s) ○ Formula ● Script □ Show browser

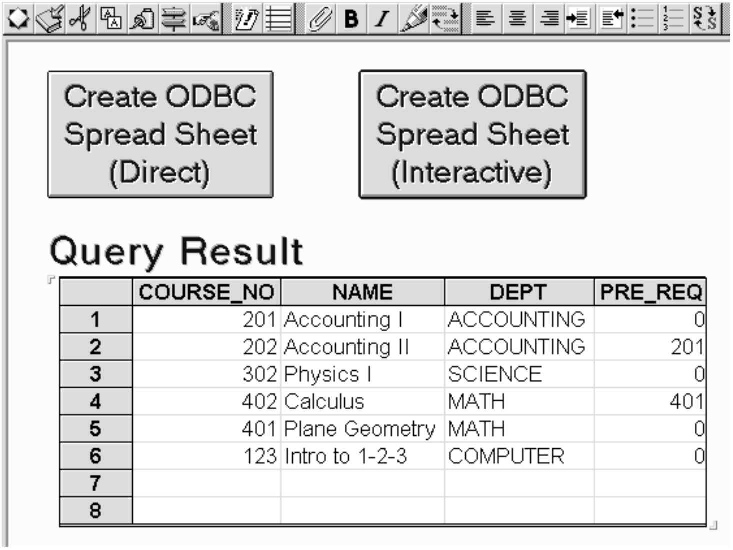
```
Set doc = udoc.Document
'** Move the cursor to the field in which a  spreadsheet
Call udoc.gotofield("RText")
Set obj = udoc.createObject("Sheet","Lotus.SpreadSh
Call obj.ODBCConnect(dbName,True,ret)
```

Scripts

```
Sub Click(Source As Button)
     Dim ws As New NotesUIWorkspace
     Dim udoc As NotesUIDocument
     Dim doc As NotesDocument
     Dim obj As Variant
     Dim dbName As String
     Dim sql As String
     Dim ret  As  Integer
     Dim CName As Long, CFormat As Long
     Dim CWidth As Long, CMaxRC As Long
     Set udoc = ws.CurrentDocument
     Set doc = udoc.Document
     '** Move the cursor to the field in which a spreadsheet
     is created.
     Call udoc.gotofield("RText")
     Set obj =
udoc.createObject("Sheet","Lotus.SpreadSheet.1")
     dbName = "dBaseDB1"
     Call obj.ODBCConnect(dbName,True,ret)
     sql = "SELECT * from courses"
     CName = True
     CFormat = True
     CWidth = True
     CMaxRC = True
     Call obj.ODBCQuery(sql, 1, 1, False, CName, CFormat,
CWidth, CMaxRC, ret)
End Sub
```

How It Works

When you click on the left button, you will get the following result: A spreadsheet is created in the rich text item of the form, and the column titles of that sheet are replaced with the column names of the queried database table (instead of “A,” “B,” etc.). If the query result is too large to fit in the visible part of the sheet window, scroll bars are displayed, provided that you specified the appropriate option on the InfoBox for the Lotus Spreadsheet component.


Create ODBC Spread Sheet (Direct)
Create ODBC Spread Sheet (Interactive)

Query Result

	COURSE_NO	NAME	DEPT	PRE_REQ
1	201	Accounting I	ACCOUNTING	0
2	202	Accounting II	ACCOUNTING	201
3	302	Physics I	SCIENCE	0
4	402	Calculus	MATH	401
5	401	Plane Geometry	MATH	0
6	123	Intro to 1-2-3	COMPUTER	0
7				
8				

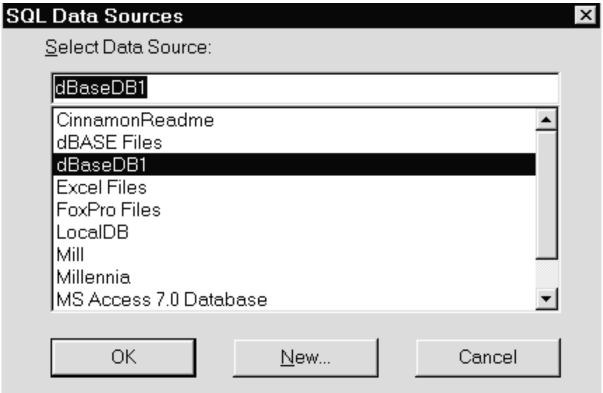
Example: An Interactive Query

Scripts

```
Sub Click(Source As Button)
     Dim ws As New NotesUIWorkspace
     Dim uDoc As NotesUIDocument
     Dim doc As NotesDocument
     Dim obj As Variant
     Dim dbName As String
     Dim sql As String
     Dim ret  As  Integer
     Dim CName As Long, CFormat As Long
     Dim CWidth As Long, CMaxRC As Long
     Set udoc = ws.CurrentDocument
     Set doc = udoc.Document
     '** Move the cursor to the field in which a spreadsheet
     is created.
     Call udoc.gotofield("RText")
     Set obj =
udoc.createObject("Sheet","Lotus.SpreadSheet.1")
     Call obj.ODBCConnect(dbName,True,ret)
     CName = True
     CFormat = True
     CWidth = True
     CMaxRC = True
     Call obj.ODBCQuery(sql, 1, 1, False, CName, CFormat,
     CWidth, CMaxRC, ret)
     End Sub
```

How It Works

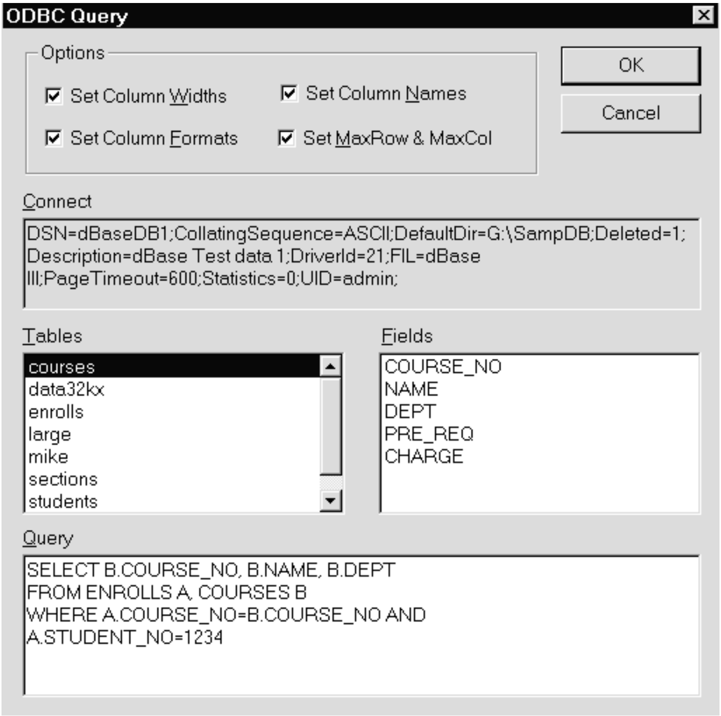
When you click on the right button, the following dialog box is displayed prompting you to specify a database for the query:



SQL Data Sources

Select Data Source: dBaseDB1

Then, the next dialog box prompts you to specify an SQL statement. As soon as you select a table in the Table listbox, all column names in the table are displayed in the Fields listbox.



Options

☑ Set Column Widths ☑ Set Column Names

☑ Set Column Formats ☑ Set MaxRow & MaxCol

Connect

DSN=dBaseDB1;CollatingSequence=ASCII;DefaultDir=G:\SampDB;Deleted=1;Description=dBase Test data 1;DriverId=21;FIL=dBase III;PageTimeout=600;Statistics=0;UID=admin;

Tables

courses
data32kx
enrolls
large
mike
sections
students

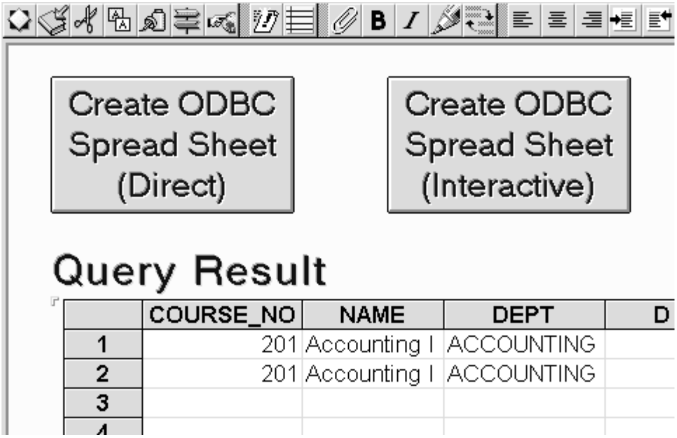
Fields

COURSE_NO
NAME
DEPT
PRE_REQ
CHARGE

Query

```
SELECT B.COURSE_NO, B.NAME, B.DEPT
FROM ENROLLS A, COURSES B
WHERE A.COURSE_NO=B.COURSE_NO AND
A.STUDENT_NO=1234
```

The following figure shows the result of the above query:


Create ODBC Spread Sheet (Direct)
Create ODBC Spread Sheet (Interactive)

Query Result

	COURSE_NO	NAME	DEPT	D
1	201	Accounting I	ACCOUNTING	
2	201	Accounting I	ACCOUNTING	
3				
4				

Chapter 17
Accessing Notes From Relational Database Management Systems and Query Tools

This chapter describes how to access Notes data from Relational Database Management Systems and query tools.

When you have completed this chapter, you should know:

- What NotesSQL is
- Relationships between RDBMS and Notes databases
- How to execute SQL against Notes data

What Is NotesSQL?

NotesSQL™ is the Lotus Notes ODBC driver for Windows, which enables ODBC-compliant DBMSs and data query tools to access, query and report on Notes-based information. Application developers have access to the Notes data store from external DBMS applications, and have query tools available which allow them to take advantage of the value of the data stored in Notes.

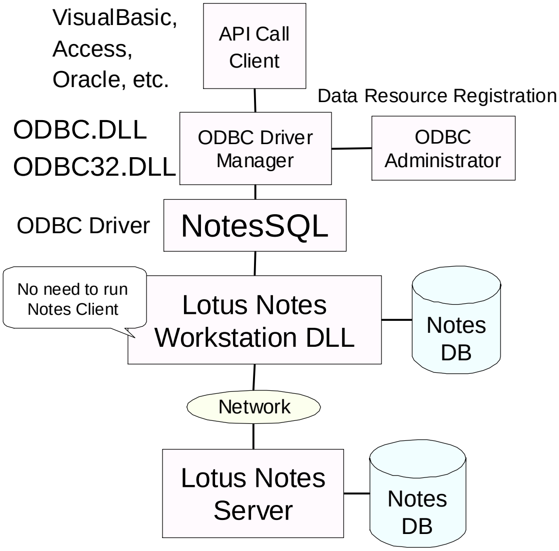
NotesSQL makes Notes-based information seamlessly available to SQL tools and applications. NotesSQL makes Notes “look” like another relational back-end data source to the SQL tool or application interface, by producing result sets that mirror the standard relational model. This allows developers working primarily with relational tools to tap the value of data stored in Notes databases.

Technical Advantages

NotesSQL allows the developer to issue SQL statements against Notes databases, which is a significant advantage to developers who wish to use Notes data in their applications. In essence, NotesSQL is really an SQL API to Notes, with full level I ODBC 2.0 compliance and level II extensions.

Structure

The way a target database is accessed from an ODBC client, such as a Notes database from Visual Basic, is the same as for any other ODBC driver. The following figure shows how to reach a Notes database from an application which uses NotesSQL.



When to Use NotesSQL

If you want to access a Notes database from an RDBMS or from another application development tool, such as Oracle or Visual Basic, use NotesSQL.

NotesSQL is designed for query and reporting tools and other ODBC-compliant DBMSs and tools to access Notes data. For example, users often need reports that incorporate data from both Notes and a DBMS. A sales force automation application can use Lotus Notes to capture information from field sales, such as customer feedback, contact management, and sales forecasts, while customer orders are often managed by DBMSs.

NotesSQL allows an external DBMS or query tool to perform table joins or combine the data from both sources in the same report. The date of last contact from the Notes sales force application could be combined with the latest customer order date via NotesSQL to produce a report on the length of the sales cycle at a customer site or across a customer set. Query tools leveraging NotesSQL provide structured data analysis of sales forecast information stored in Lotus Notes. Similarly, that same information collected in Notes can be pulled into DB2, for central storage and distribution.

If you already have an ODBC-compliant program to access an RDBMS, you may also be able to access a Notes database without any modification of your program. In this case, the only thing you have to worry about is the difference between an RDBMS and a Notes database, such as data types, the conformance level of SQL, and so on, as a Notes database is not a relational database.

Functionality

ODBC Conformance Level of NotesSQL

There are three conformance levels specified in the ODBC API, which are Core Level, Extension Level 1 and Extension Level 2. These are just general guidelines, and not all the drivers available today support all three levels. Moreover, even if an ODBC driver supports a conformance level, not all the APIs of that level are always supported. Many ODBC drivers conform to both the Core Level and Extension Level 1.

NotesSQL supports all three levels of conformance except for some APIs of Extension Level 2. This is described in the Limitations section later in this chapter.

Core Level

Core Level is a minimum function set of the ODBC specifications. It mainly supports:

- Allocation and deallocation of the environment
- Connection and disconnection of a database
- SQL preparation and execution
- Fetch data
- Transaction control

Note This is not supported in NotesSQL. Lotus Notes does not have a transaction mechanism.

Extension Level 1

Extension Level 1 extends the Core Level function set. It mainly supports:

- Retrieving the table schema
- Connecting to a database interactively
- Getting and putting data of a result set

Extension Level 2

Extension Level 2 contains more sophisticated functions. It mainly supports:

- Primary key and foreign key

  Note This is not supported in Notes, and thus, not in NotesSQL.

- Tapping connection

The following three features are not supported in NotesSQL:

- Table and column privilege control
- Stored procedure
- Cursor control

SQL Grammar Conformance Level of NotesSQL

The following three levels are available:

Minimum SQL

Minimum SQL only supports the character data type and simple operations.

DDL	*DCL*	*DML*	*Expressions*	*Data Types*

CREATE TABLE
DROP TABLE
		SELECT
		INSERT
		UPDATE Searched
		DELETE Searched
			Numeric Operations (+, -, *, / , <, >, <=, >=, =, <>)	CHAR
				VARCHAR
				LONGVARCHAR

Core SQL

Core SQL supports DCL (Data Control Language), and many operations and data types.

DDL	*DCL*	*DML*	*Expressions*	*Data Types*
ALTER TABLE CREATE INDEX CREATE VIEW DROP INDEX DROP VIEW	GRANT REVOKE	SELECT full syntax	Subselect Aggregation (SUM, MIN, MAX, AVG, COUNT)	DECIMAL NUMERIC SMALLINT INTEGER REAL FLOAT DOUBLE

Extended SQL

Extended SQL supports advanced operations, such as cursor related operations and outer join.

DDL	*DCL*	*DML*	*Expressions*	*Data Types*
		UPDATE Positioned DELETE Positioned Outer Join Cursor Control SELECT FOR UPDATE	Scalar functions	TINYINT, BIGINT, BINARY, VARBINARY, LONG, BIT, DATE, TIME, TIMESTAMP

Note NotesSQL supports Minimum SQL and some of the other SQL grammar.

Checking the Conformance Level

If you want to check the conformance levels of your ODBC driver, such as NotesSQL, the SQLGetInfo ODBC function can help you do this. The SQLGetInfo function is included in Extension Level 1 as an ODBC API conformance level.

NotesSQL supports the SQLGetInfo function.

Constant and Function Declarations in Visual Basic

In most programming environments, such as Visual Basic, you need to describe function declare statements in the ODBC DLL to use the NotesSQL functions.

In ODBC programs, SQLAllocEnv, SQLAllocConnect, SQLConnect, and SQLAllocStmt are often essential. If required, you can also use SQLDriverConnect and SQLError.

```
'** Constant Declarations for SQLGetInfo
'** to check ODBC API and ODBC SQL Conformance Level
Public Const SQL_ODBC_API_CONFORMANCE As Long = 9
Public Const SQL_ODBC_SQL_CONFORMANCE As Long = 15
'** For ODBC API Conformance Level
Public Const SQL_OAC_NONE As Long = 0
Public Const SQL_OAC_LEVEL1 As Long = 1
Public Const SQL_OAC_LEVEL2 As Long = 2
'** For ODBC SQL Conformance Level
Public Const SQL_OSC_MINIMUM As Long = 0
Public Const SQL_OSC_CORE As Long = 1
Public Const SQL_OSC_EXTENDED As Long = 2
```

```
'** Options for SQLDriverConnect
Public Const SQL_DRIVER_NOPROMPT As Long = 0
Public Const SQL_DRIVER_COMPLETE As Long = 1
Public Const SQL_DRIVER_PROMPT As Long = 2
Public Const SQL_DRIVER_COMPLETE_REQUIRED As Long = 3
'** Return Code
Public Const SQL_ERROR As Long = -1
Public Const SQL_INVALID_HANDLE As Long = -2
Public Const SQL_NO_DATA_FOUND As Long = 100
Public Const SQL_SUCCESS As Long = 0
Public Const SQL_SUCCESS_WITH_INFO As Long = 1
'** ODBC Functions to issue SQLGetInfo
'** To get an Environment Handle
Declare Function SQLAllocEnv Lib "odbc32.dll" (phenv&) As
Integer
'** To get a Connection Handle
Declare Function SQLAllocConnect Lib "odbc32.dll" (ByVal
henv&, phdbc&) As Integer
'** To establish a connection
Declare Function SQLConnect Lib "odbc32.dll" (ByVal hdbc&,
ByVal szDSN$, ByVal cbDSN%, ByVal szUID$, ByVal cbUID%, ByVal
szAuthStr$, ByVal cbAuthStr%) As Integer
'** To establish a connection with a dialog box
Declare Function SQLDriverConnect Lib "odbc32.dll" (ByVal
hdbc&, ByVal hWnd As Long, ByVal szCSIn$, ByVal cbCSIn%,
ByVal szCSOut$, ByVal cbCSMax%, cbCSOut%, ByVal fDrvrComp%)
As Integer
'** To get a Statement Handle
Declare Function SQLAllocStmt Lib "odbc32.dll" (ByVal hdbc&,
phstmt&) As Integer
'** To get information on an ODBC driver
Declare Function SQLGetInfo Lib "odbc32.dll" (ByVal hdbc&,
ByVal fInfoType%, ByRef rgbInfoValue As Any, ByVal
cbInfoMax%, cbInfoOut%) As Integer
'** To get error information
Declare Function SQLError Lib "odbc32.dll" (ByVal henv&,
ByVal hdbc&, ByVal hstmt&, ByVal szSqlState$, pfNativeError&,
ByVal szErrorMsg$, ByVal cbErrorMsgMax%, pcbErrorMsg%) As
Integer
```

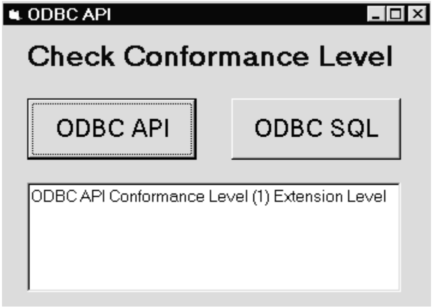
How to Issue SQLGetInfo in Visual Basic

The following program is about getting information on the ODBC API Conformance level. If you don't need a dialog box to specify a data source name, you can replace SQLDriverConnect with SQLConnect.

```
Private Sub Command1_Click()
Dim ret As Integer
Dim msg As String
Dim rInfo As Long
Dim rSize As Integer
Dim connect As String * 255
Dim connectLen As Integer
Dim henv As Long, hdbc As Long, hstmt As Long
'** To get an Environment Handle
ret = SQLAllocEnv(henv)
If ret = SQL_ERROR Then
  Call ErrorMSG(henv, hdbc, hstmt)
  Exit Sub
End If
'** To get a Connection Handle
ret = SQLAllocConnect(henv, hdbc)
If ret = SQL_ERROR Then
  Call ErrorMSG(henv, hdbc, hstmt)
  Exit Sub
End If
'** To establish a Connection
ret = SQLDriverConnect(hdbc, Me.hWnd, "", 0, connect,
Len(connect), connectLen, SQL_DRIVER_PROMPT)
If ret = SQL_ERROR Then
  Call ErrorMSG(henv, hdbc, hstmt)
  Exit Sub
End If
'** To get a Statement Handle
ret = SQLAllocStmt(hdbc, hstmt)
If ret = SQL_ERROR Then
  Call ErrorMSG(henv, hdbc, hstmt)
  Exit Sub
End If
'** To get ODBC API Conformance Level
ret = SQLGetInfo(hdbc, SQL_ODBC_API_CONFORMANCE, rInfo, 300,
rSize)
If ret = SQL_ERROR Then
  Call ErrorMSG(henv, hdbc, hstmt)
  Exit Sub
End If
Select Case rInfo
Case SQL_OAC_NONE:   msg = "Core Level"
Case SQL_OAC_LEVEL1: msg = "Extension Level 1"
Case SQL_OAC_LEVEL2: msg = "Extension Level 2"
End Select
Text1.Text = "ODBC API Conformance Level (" & rInfo & ") " &
msg
End Sub
```

How it Works

The following figure shows how the above Visual Basic program works. When you click the ODBC API button, the ODBC API Conformance Level is displayed in a text box. The SQL Data Sources dialog box is displayed for you to specify a data source name, since SQLDriversConnect is issued without any data source names to establish a connection in the above program.



ODBC API Conformance Level (1) Extension Level

NotesSQL Conformance Level Limitations

The following sections describe NotesSQL limitations with regard to Conformance level support.

Data Mapping Limitations

Fields and columns containing the following @Functions cannot be queried by SQL in NotesSQL:

- @All
- @DeleteDocument
- @DeleteFields
- @DocChildren
- @DocLevel
- @DocNumber
- @DocParentNumber
- @DocSiblings
- @Error
- @IsCategory
- @IsExpandable
- @Unavailable

SQL Limitations

NotesSQL conforms to the Minimum SQL Level, most of the Core SQL Level and some of the Extended SQL Level. But there are some limitations of SQL usage as follows:

- NULL and NOT NULL are not supported in ALTER TABLE.
- UNIQUE is not supported in CREATE INDEX.
- NULL, NOT NULL, UNIQUE PRIMARY KEY and REFERENCES are not supported in CREATE TABLE.
- GRANT and REVOKE are not supported.
- Parameters are supported only in INSERT, DELETE and SELECT.

ODBC API Limitations

The following ODBC APIs are not supported in NotesSQL. Access privileges to a database are limited only to access controls in a Notes database.

- SQLColumnPrivileges
- SQLTablePrivileges
- SQLForeignKeys
- SQLPrimaryKeys
- SQLProcedureColumns
- SQLProcedures

The following ODBC APIs in NotesSQL have different implementations of ODBC specifications:

- SQLCancel
- SQLColumns
- SQLGetConnectOption
- SQLSetConnectOption
- SQLGetStmtOption
- SQLSetStmtOption
- SQLSetParam
- SQLSetScrollOptions
- SQLTables
- SQLTransact

Software Requirements

The software requirements are the same as those required by any other ODBC driver and ODBC-compliant system:

- Lotus Notes Client or Server
- NotesSQL Release 2.0 or higher
- ODBC Driver Manager Release 2.0 or higher

Note You must choose either the 16-bit or the 32-bit version of both NotesSQL and the ODBC Driver Manager as required by your application.

Mapping Resources Between an RDBMS and a Notes Database

Connection String

When you connect to a data source, you need to supply a connection string, which indicates a database name, a server name and so on.

• DSN	Data Source Name registered in ODBC Driver Manager
• Database	Notes DB Name
• Server	Notes Server Name or Blank for Local Notes DB

The user ID is supplied by the NOTES.INI file which belongs to the Notes workstation or the Notes server.

The Notes system rather than the ODBC driver (NotesSQL) prompts for the password, so that Notes security is maintained.

Table and View

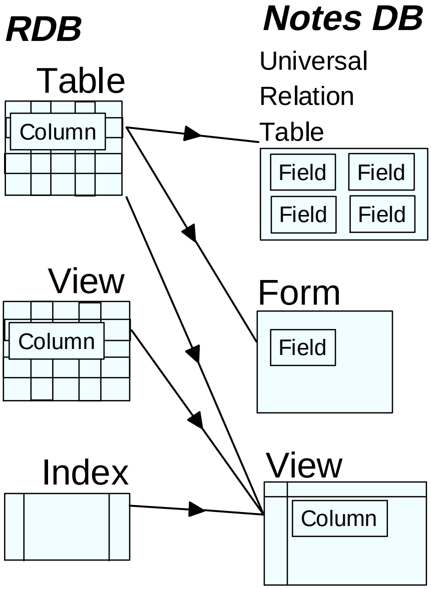
A somewhat different concept is required to understand the relationship between tables and views with regard to a Notes database and an RDBMS.

NotesSQL can deal with a Notes database as one real table, which is called Universal Relation Table in NotesSQL. The Universal Relation Table holds all fields contained in a Notes database.

Note The Universal Relation table name is the same as the database name.

The only supported SQLs are SELECT and CREATE VIEW. Explicit field names must be specified in a SELECT statement, rather than using 'SELECT *'. All fields are handled as text data.

The following figure shows how a form and a view in a Notes database are mapped to a table, a view, and an index in an RDBMS.



Note View names in a Notes database must be different from any form names in that Notes database, because both of them can be mapped to a table in an RDB.

A Notes View as an Index of an RDB

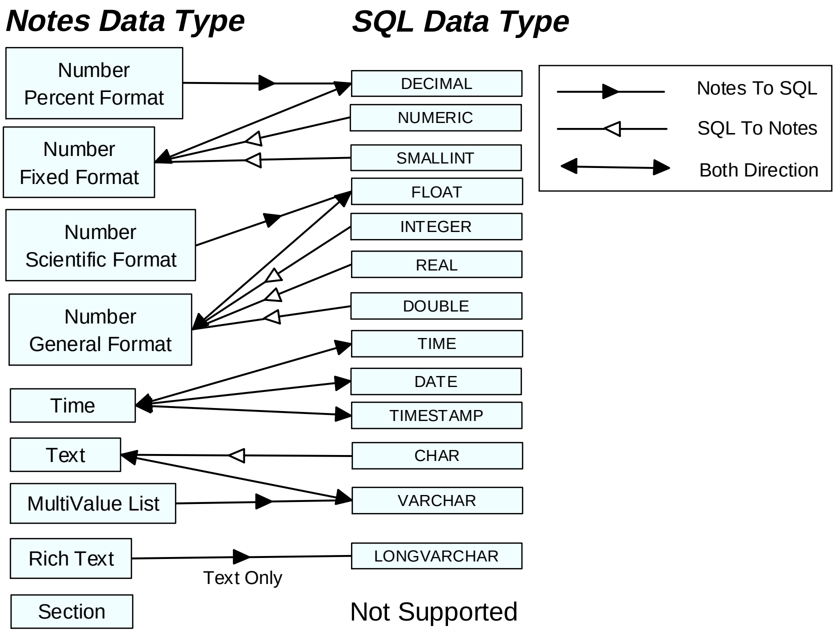
A Notes view can be handled as both a table and a view in NotesSQL. When the following criteria in a design of a Notes view are satisfied, a Notes view is mapped to an index in NotesSQL

- Either ‘SELECT Form=FormName’ or ‘SELECT @ALL’ in a selection formula
- Only simple field references in columns and no formulas
- At least one sorted column

Mapping Data Types Between an RDB and a Notes Database

The following figure shows how specific data types are mapped in a Notes database and an RDB.

For example, a Percent Format Number field in a Notes database is converted to DECIMAL data in an SQL RDB. NUMERIC data in an SQL RDB is converted to a Fixed Format Number field in a Notes database.



A List Field (Multi Value Field) is only available in some DMLs, such as SELECT, INSERT, UPDATE and DELETE. The only supported data type is Text. The representation is like this: ‘String1;String2’.

A rich text field cannot be created by NotesSQL. Only text portions in a rich text field can be retrieved.

Basic API Calling Sequences

The following ODBC API sequences are necessary to issue an SQL statement and to retrieve a result in your RDB or development tool. Query tools perfom these statements automatically for the user.

1. **SQLAllocEnv**

   To obtain an Environment Handle.

   An Environment Handle must be supplied to any SQLAllocConnect APIs.

2. **SQLAllocConnect**

   To obtain a Connection Handle.

   A Connection Handle must be supplied to any SQLConnect APIs.

3. **SQLConnect**

   To connect to a database source.

4. **SQLAllocStmt**

   To obtain a Statement Handle.

   A Statement Handle must be supplied to many APIs to deal with table and column information and to issue an SQL statement, such as SQLColumns, SQLExecDirect, SQLFetch, SQLData, and so on.

5. **SQLExecDirect**

   To run an SQL statement.

6. **SQLFetch**

   To obtain a result set of an SQL.

7. **SQLGetData**

   To retrieve column data from a result set.

8. **SQLFreeStmt**

   To discard a Statement Handle.

9. **SQLDisconnect**

   To close a connection.

10. **SQLFreeConnect**

    To discard a Connection Handle.

11. **SQLFreeEnv**

    To discard an Environment Handle.

Example: Accessing Notes From Visual Basic

Visual Basic has some ways of accessing Lotus Notes using the ODBC feature:

- **Remote Data Control** (RDC)

  This is a visual control to deal with remote data access. Basically, no programming is needed to access Notes. This feature can provide read and write access to a database, but it is often only used to retrieve database data.

- **Remote Data Object**

  Some methods and properties are available in the Remote Data Object to access an ODBC database.

- **ODBC API Call**

  Environment handles, Connection handles and statement handles are retrieved in both the Remote Data Control and the Remote Data Object. They can be combined with each other.

Program Structure

The following sections provide some detailed information on how our sample application accesses a Notes database from Visual Basic.

Creating a Data Source List

When the form module of our example is loaded, a data source list is created by the ListDataSources subroutine and listed in a listbox as follows:

```
'** Form_Load is executed, when a Form is loaded.
Private Sub Form_Load()
'** To create a data source list using ODBC API
Call ListDataSources
End Sub
'** ListDataSources can make a data source list
'** and display it in a listbox
Sub ListDataSources()
Dim ret As Integer
Dim dataSource As String * 32
Dim dsDesc As String * 2048
Dim dsLen As Integer, dsDescLen As Integer
Dim henv As Long, hdbc As Long, hstmt As Long
'** To get an Environment Handle
ret = SQLAllocEnv(henv)
If ret = SQL_ERROR Then
  Call ErrorMSG(henv, hdbc, hstmt)
  Exit Sub
End If
```

```
'** To get Data Source List
'** Fetch the First Record
ret = SQLDataSources(henv, SQL_FETCH_FIRST, dataSource, 31,
dsLen, dsDesc, 2047, dsDescLen)
Do
  If ret = SQL_ERROR Then
    Call ErrorMSG(henv, hdbc, hstmt)
    Exit Sub
  ElseIf ret = SQL_NO_DATA_FOUND Then
'** When the end of records, Exit Do-sLoop
    Exit Do
  End If
  list1.AddItem dataSource
'** Fetch a Next Record
  ret = SQLDataSources(henv, SQL_FETCH_NEXT, dataSource, 31,
dsLen, dsDesc, 2047, dsDescLen)
Loop
End Sub
```

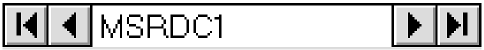
Creating a Table List

To create a table list according to the data source name you specified in the data source list, click the Get Table List button. It is easy to get information about table names in a database using one of the RDO properties (rdoTables) as follows.

```
'** This subroutine is executed, when a button is clicked
Private Sub Command2_Click(
Dim tbNum As Integer
Dim tables As Variant
MSRDC1.SQL = ""
MSRDC1.DataSourceName = text2.Text
'** To connect to a data source
MSRDC1.Refresh
List2.Clear
'** To get table names in a data source and set them to a
listbox
For Each table In MSRDC1.Connection.rdoTables
  List2.AddItem tables.Name
Next
End Sub
```

How to Issue an SQL Statement and Get a Result Set

Before an SQL statement is executed, you need to create the following Remote Data Control anywhere in your Visual Basic form. It should be invisible using a property. To execute an SQL statement with this control, specify a data source name in the DataSourceName property and an SQL statement in the SQL property.



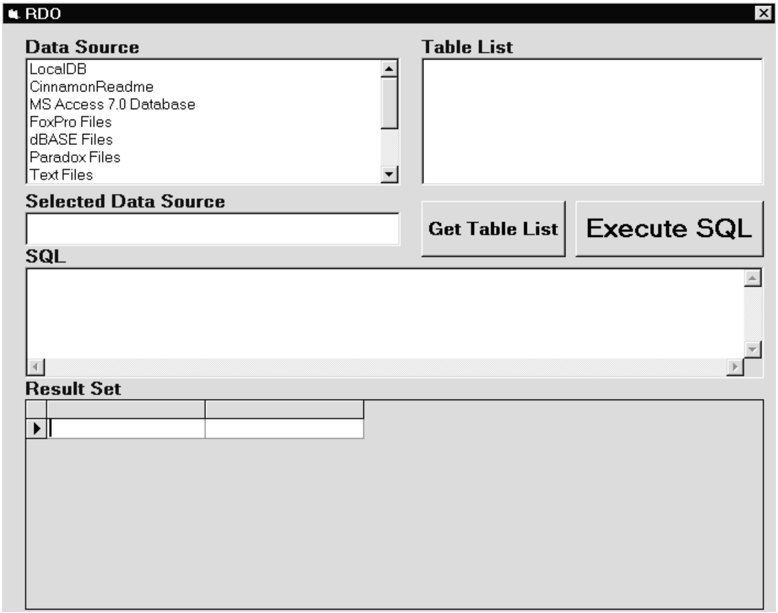
```
'** This subroutine is executed, when a button is clicked
Private Sub Command1_Click()
'** To Ignore Run Time Error
On Error Resume Next
MSRDC1.DataSourceName = text2.Text
MSRDC1.SQL = Text1.Text
'** To issue a query
MSRDC1.Refresh
'** Cancel To Ignore Run Time Error
On Error GoTo 0
End Sub
```

Note Our sample program does not provide for error handling, as we just intended to show some behaviors of NotesSQL. When you input a wrong SQL statement, nothing will happen. If you want to create a fully developed error handling routine, remove the On Error statements in the above program.

How it Works

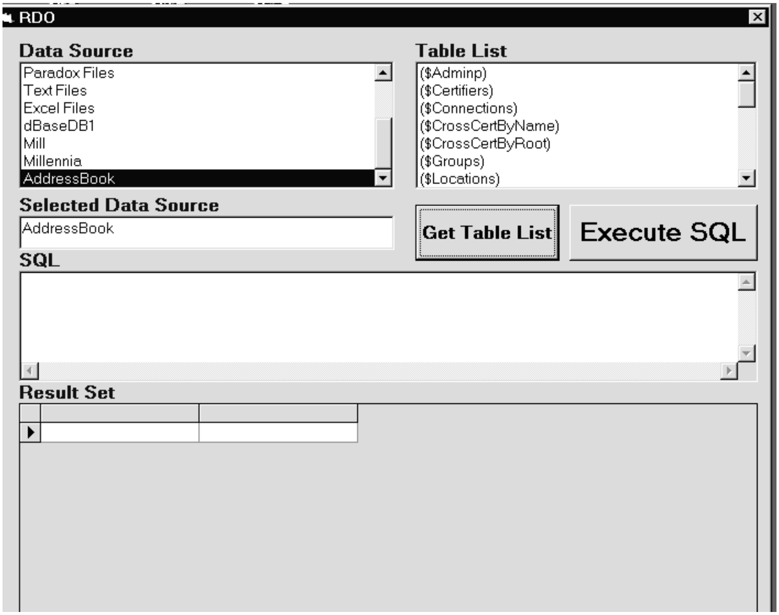
This is how it works:

1. After the program is launched, the following data source list is displayed. The data source list is only created at launch time.



LocalDB
CinnamonReadme
MS Access 7.0 Database
FoxPro Files
dBASE Files
Paradox Files
Text Files

2. Choose a data source from the list and click the Get Table List button. This displays a table list on the right-hand side of the dialog box. The data source name which you chose is also displayed in the Selected Data Source field. The table list is only used to refer to table names in a database and has no effect on the following SQL query.



RDO

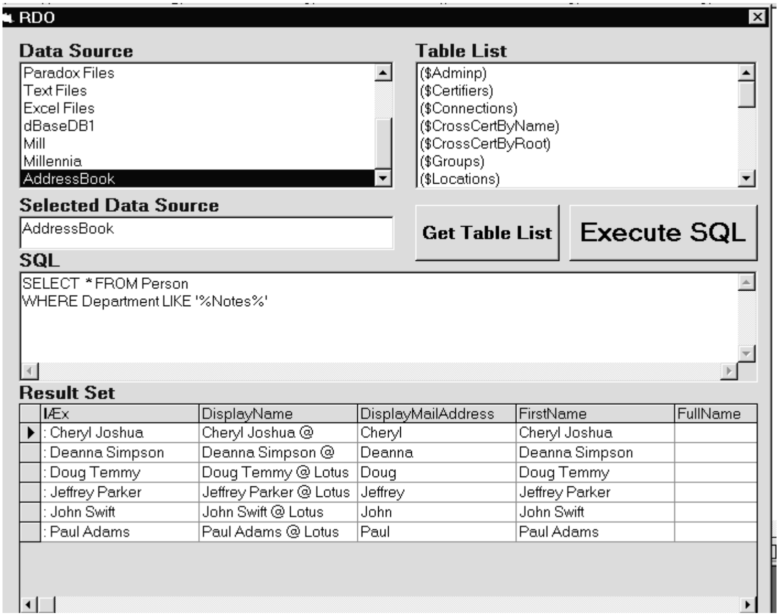
Data Source: Paradox Files, Text Files, Excel Files, dBaseDB1, Mill, Millennia, AddressBook

Table List: ($Adminp), ($Certifiers), ($Connections), ($CrossCertByName), ($CrossCertByRoot), ($Groups), ($Locations)

Selected Data Source: AddressBook

3. To get an SQL result set, specify an SQL statement in the SQL text box and click the Execute SQL button.

   The following figure shows a result set of a query to an Address Book in Lotus Notes using the NotesSQL ODBC driver.



Selected Data Source: AddressBook

```
SELECT  * FROM Person
WHERE Department LIKE '%Notes%'
```

	IÆx	DisplayName	DisplayMailAddress	FirstName	FullName
▶	: Cheryl Joshua	Cheryl Joshua @	Cheryl	Cheryl Joshua	
	: Deanna Simpson	Deanna Simpson @	Deanna	Deanna Simpson	
	: Doug Temmy	Doug Temmy @ Lotus	Doug	Doug Temmy	
	: Jeffrey Parker	Jeffrey Parker @ Lotus	Jeffrey	Jeffrey Parker	
	: John Swift	John Swift @ Lotus	John	John Swift	
	: Paul Adams	Paul Adams @ Lotus	Paul	Paul Adams	

Chapter 18
High Volume Data Transfer With NotesPump 2.0

This chapter describes Lotus NotesPump™ Release 2.0, which is used to transfer data between Notes databases and external databases on different types of Database Management Systems. The NotesPump LSX is introduced and a sample is described. This chapter is based on NotesPump Release 2.0 Beta 3. Actual functionality and screen definitions may change at the final release.

To get further information about NotesPump, visit the Lotus Web site at: http://www.lotus.com/edgehome.

About Lotus NotesPump

Lotus NotesPump is a server-based data transfer engine that enables high performance, scalable exchange of data between relational database management systems (RDBMS) and Notes databases. Using NotesPump, developers, database administrators, and Information Technology professionals can establish frequently scheduled, event-driven, or ad-hoc bi-directional exchanges of data among corporate data store systems, including IBM DB2®, Oracle®, Sybase® and ODBC™ sources, and Notes databases.

NotesPump comprises two components: the server engine, and administration databases. The server engine operates on OS/2 Warp 3.0, Windows NT 3.51, and HP-UX™ 10.01 platforms, and may optionally be installed on a Notes Server machine as a co-resident task.

The administrative databases consist of the NotesPump Administrator, a Notes database used to store instruction tasks executed by the server engine(s), and the NotesPump Log which records NotesPump server processing results in a Notes database.

The NotesPump server executes specific transfer operations including Direct Transfer, Replication, Polling (event monitoring), Command (executes command line and database instructions), and Scripted (utilizes LotusScript®).

NotesPump Enterprise Features

NotesPump was designed to address corporate requirements to move high-volume data sets between dissimilar databases, and to be managed within enterprise network infrastructures. Lotus NotesPump provides high performance options which enhance data transfer performance within these system environments:

- 32-bit multi-tasking server engine

  NotesPump Operates as a multi-tasking processor able to handle multiple data transfer operations concurrently.

- Scalable component architecture

  Consisting of server engines, database modules, and defined transfer Activities, may be added when required via the NotesPump management application. NotesPump continually monitors the network system for new components, and newly-added resources do not interrupt service.

- SNMP support

  Servers may be monitored by SNMP management systems.

- Customized, high performance database Link Options

  Link Options are specific to individual databases supported to provide higher speed data transfer throughput. Customized options include Array processing, Bulk Transfer and database commit and rollback options.

- DB2 Link support

  A native interface to IBM DB2 databases on all supported platforms.

- Data Propagator relational support

  Custom NotesPump Activity enables propagation of IBM Data Propagator relational Consistent Changed Data table information (CCD) to NotesPump-supported targets.

- Notes R4 Administrator

  Leverages Notes R4 ease-of-use features including Navigator and collapsible form sections. New database browsing capabilities facilitate easier management of server task definitions.

- LotusScript support

  The NotesPump LSX makes NotesPump classes available for scripting using the Notes R4 client Integrated Development Environment (IDE). Using Scripting within NotesPump, administrators can customize data transfer applications processed by the NotesPump server.

- HP-UX 10.01 NotesPump server support

- Internet Web browser access

  Permits Web clients to submit and receive data queries processed via the NotesPump server.

- International data set character translation

  Automatic detection and translation of international database character sets.

- NotesPump API

  A C-based API used to create custom database links or Activities to operate with the NotesPump server. The NotesPump API broadens the scope of databases and applications that can be used with the NotesPump server system.

Functions

NotesPump offers the following functions:

- Data migration on a scheduled basis among Notes, DBMS or File sources

  The movement of data is either scheduled, polled or manually initiated. NotesPump servers are administrated by a Notes application, the NotesPump Administrator (see examples later in this section). A snapshot of the data is taken and copied to the target platform on a periodic basis. Selection criteria are available to control the amount of data copied.

- Replication services to synchronize data source information

  NotesPump server operations include the capability to replicate dissimilar database sources, thereby maintaining synchronized information in each. Using a Primary key common to each database, Replication Activities may be scheduled to periodically evaluate and exchange data, keeping information in each consistent and current. Use of a time-stamp key refines this operation, synchronizing only new updates between the two data stores.

- Conditional processing

  NotesPump classes may be accessed from the Lotus Notes Release 4 integrated development environment (IDE) to create conditional processing routines processed by the NotesPump server, thereby extending database interchange capabilities.

- Event monitoring

  Polling allows the NotesPump user to define conditions to monitor in Notes or DBMS sources, such as an insert to an DBMS table or a Notes database. When a condition is satisfied, NotesPump immediately initiates a specified data exchange Activity to accomplish data exchange. Optionally, system administrators may be notified of processing results via Notes Mail, and view the NotesPump server processes from an SNMP management station.

- Internet data queries

  The NotesPump CGI program enables Web-based clients to query NotesPump supported database sources.

NotesPump Applications

Data distribution

In the data distribution area, for instance, work order data volumes generated from a customer service center maintained in a mainframe DBMS may be transferred via the NotesPump server to field-based technical staff using Lotus Notes. This is particularly useful to occasionally connected users who do not have continuous network access to the DBMS server. Conditions for transfer to specific staff regions may be specified through NotesPump, providing rapid and frequent updates of specified information from the customer service center to appropriate company technical staff locations.

Data synchronization between dissimilar data sources

A corporation might maintain current and historical employee data in a Human Resources DBMS. Data updates by employees are controlled via a Lotus Notes application, permitting specified personal data changes to be propagated to the Human Resources DBMS. Using the NotesPump Replication Activity and Employee ID key common to both databases, data is synchronized between the Notes Human Resource application and the DBMS application. This ensures controlled and consistent updates to company information.

Monitor data source transactions

Using the Polling Activity, the NotesPump server can be instructed to monitor a data source for a change, such as an insert of a new sale over a certain dollar amount. When the condition is met, NotesPump instantly initiates a defined data transfer, such as propagation of the new amount to another source and generation of a notification mail message.

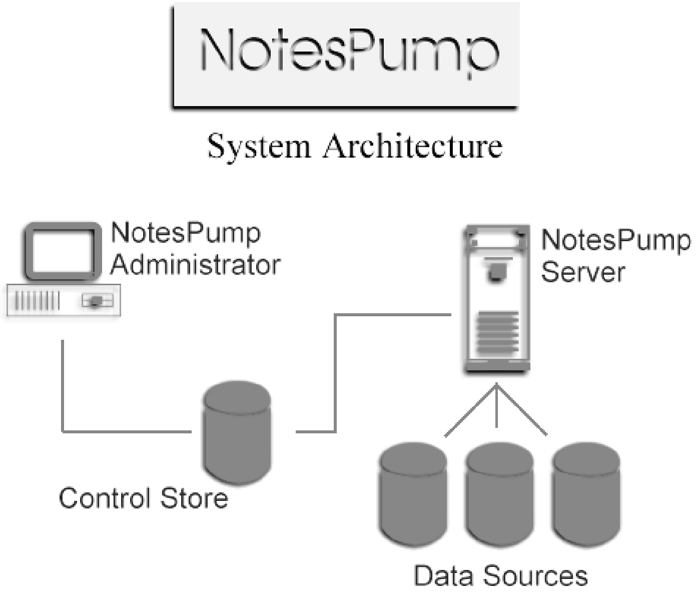
Query DBMS information from the Internet

Using NotesPump CGI program support, Web-based clients may enter their customer order number to a corporate Internet site location, and electronically learn the status of their order.

Architecture

NotesPump is organized in the following structure:

- NotesPump Server
- NotesPump Administrator
- Data Sources



NotesPump Components

NotesPump Server

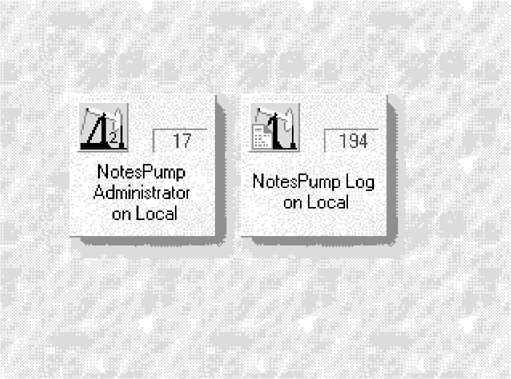
The Notes Server is the location of the NotesPump Control Store — the Administration database. It also holds the NotesPump Server's processing log database, and is the target and/or source for data transfer Activities.

NotesPump Engine

The NotesPump Engine provides the back-end interfaces to supported databases. It controls NotesPump Activities as defined in the Notes Control Store. The NotesPump Engine works as a multi-tasking data transfer pump by executing multiple data transfer tasks concurrently. The processing is monitored and logged into the NotesPump log database. Monitoring also enables the NotesPump engine to re-execute Activities upon failure. The NotesPump Engine can run on a Notes server as a co-resident process.

NotesPump Administrator

The NotesPump Administrator consists of the Notes database containing the information that NotesPump Servers need to function as well as the Notes interface used to manage that database. The database contains all the configuration, connection, and scheduling elements that NotesPump uses to collect, transfer, and write data. The information in the database is also known as the Control Store.



A single Administrator database can hold information used by several NotesPump Servers, but each NotesPump Server can be governed by only one Administrator database.

Because the Administrator is a Notes database, you update it by using the Notes application interface and navigators to create, view, and edit documents. These documents can be created through the Create menu in Notes (once the Administrator database is opened on the Notes desktop) or displayed through any of the views available in the View menu or navigator. You can think of these documents as “work orders” that tell NotesPump what to do and when.

- Configuration documents

  Define the NotesPump Administrator and the NotesPump Servers.

- Link documents

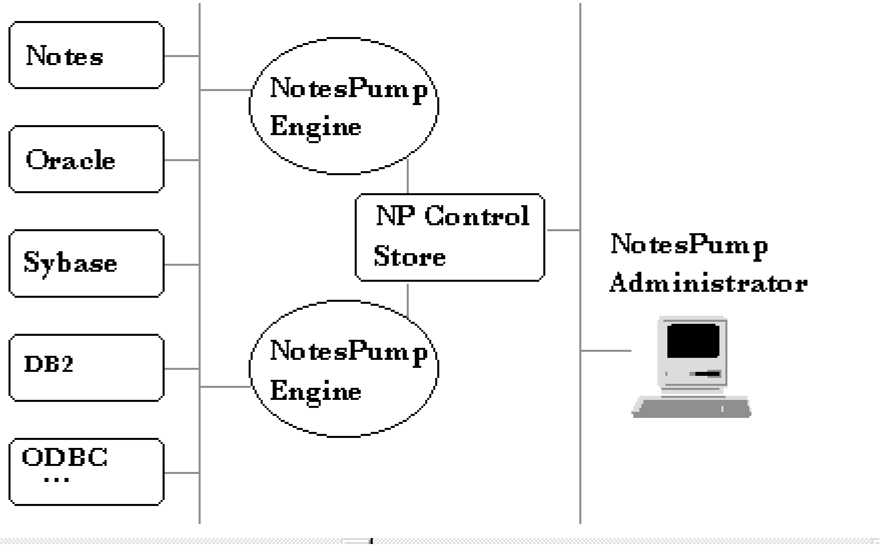
  Define the connections to the different data sources. The names and the network locations of connected databases are specified here.

- Activity documents

  Define data transfer Activities. A wide array of scheduling and data transfer options is provided with these documents.

DBMS Servers

These are the actual DBMS servers which represent the target and source for NotesPump data transfers. These NotesPump components are related as shown below:



NotesPump Installation

The NotesPump server must be installed on a machine where native Notes DLLs are already installed. The operation of NotesPump does not require that this client or server be running.

During the installation the Administrator databases are stored on the Notes server.

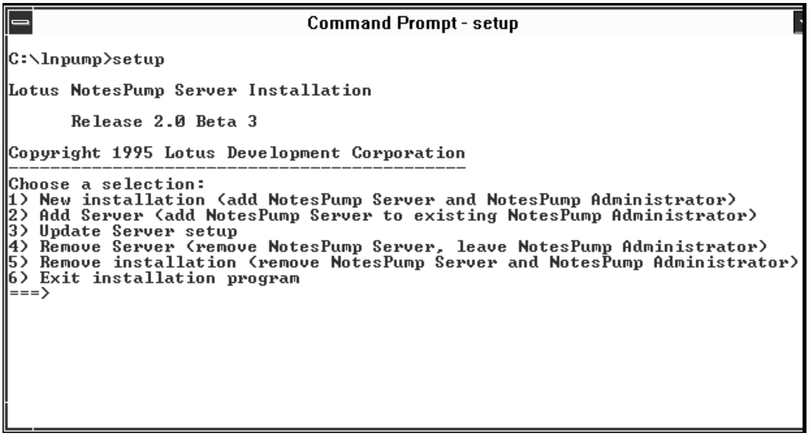
The NotesPump Server machine also requires the appropriate database communication software, such as IBM DDCS, or CAE and Communications Manager, Oracle SQL*Net, Sybase Netlib and ODBC, to be available to connect to the respective DBMS required.

Installation Steps

The following steps guide you through the installation of NotesPump Server:

1. Install the Program files from the installation media.
2. Check connectivity to your DBMS and Notes Sources and Targets by using the respective test program supplied with NotesPump.

Run the Setup Program to create the first NotesPump server and NotesPump Administrator.



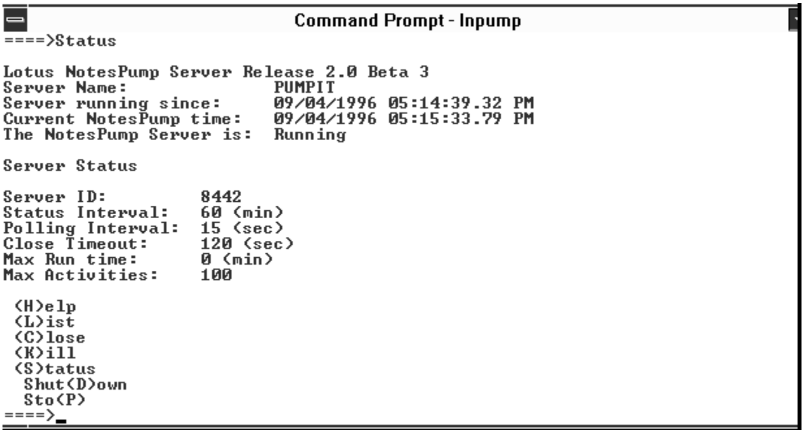
Command Prompt - setup

```
C:\lnpump>setup

Lotus NotesPump Server Installation

      Release 2.0 Beta 3

Copyright 1995 Lotus Development Corporation
--------------------------------------------
Choose a selection:
1) New installation (add NotesPump Server and NotesPump Administrator)
2) Add Server (add NotesPump Server to existing NotesPump Administrator)
3) Update Server setup
4) Remove Server (remove NotesPump Server, leave NotesPump Administrator)
5) Remove installation (remove NotesPump Server and NotesPump Administrator)
6) Exit installation program
===>
```

Start the NotesPump Server

When started, the NotesPump Server connects to the NotesPump Administrator database and immediately runs any overdue activities. It continues to poll the Administrator database, at the interval defined in the Server Configuration document, to run Activities until it is shut down.

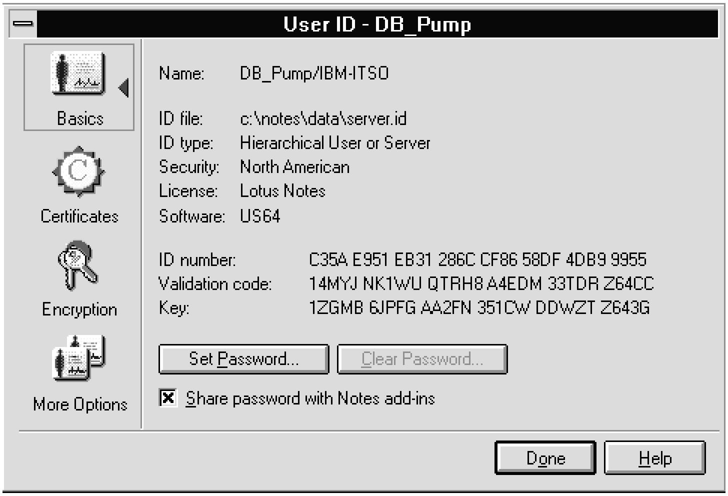
Start the NotesPump server by running lnpump.exe on a command prompt window. The program screen is shown below. Several menu selections are available to help you view Activities that are running, stop them, and also stop the server.



Command Prompt - lnpump

```
====>Status

Lotus NotesPump Server Release 2.0 Beta 3
Server Name:              PUMPIT
Server running since:     09/04/1996 05:14:39.32 PM
Current NotesPump time:   09/04/1996 05:15:33.79 PM
The NotesPump Server is:  Running

Server Status

Server ID:          8442
Status Interval:    60 (min)
Polling Interval:   15 (sec)
Close Timeout:      120 (sec)
Max Run time:       0 (min)
Max Activities:     100

 (H)elp
 (L)ist
 (C)lose
 (K)ill
 (S)tatus
  Shut(D)own
  Sto(P)
====>_
```

Tip If your Notes Server has a password, you must allow the server's User ID to share the password with add-in tasks like NotesPump. Do this by clicking on the checkbox in the File - Tools - User ID dialog box as shown below:



User ID - DB_Pump

Name: DB_Pump/IBM-ITSO

ID file: c:\notes\data\server.id
ID type: Hierarchical User or Server
Security: North American
License: Lotus Notes
Software: US64

ID number: C35A E951 EB31 286C CF86 58DF 4DB9 9955
Validation code: 14MYJ NK1WU QTRH8 A4EDM 33TDR Z64CC
Key: 1ZGMB 6JPFG AA2FN 351CW DDWZT Z643G

☒ Share password with Notes add-ins

Terminology

Before you start creating NotesPump Activities and links, it is important to clarify some of the terminology and its relationship between the DBMS relational world and the Notes unstructured design.

In Notes, the data table of a relational database is comparable to a form or view. The Notes form basically represents the definition of the data in a table, while views are row-and-column displays of the data or of a subset or superset of the data.

The relational database field or column finds its equivalent in Notes as a form field or as a view column. Records are displayable in Notes either as documents (defined by a form) or as rows in a view.

Here is a set of the terms important to work with NotesPump, and their definitions:

Column A set of similar data such as phone numbers or customer names, usually displayed in column format. A column contains the fields having the same position in the data records of a table. In a Notes view, fields appear in a column presentation.

Database A collection of metadata and data. For example, a Notes file and forms within.

Document A Notes document is a database entry that contains information. It is the equivalent of a database row and appears in row format in a view.

Field A specific kind of data represented in Notes as an item within a form or document.

Form A Notes form defines the fields and data types in a set of documents in a Notes database.

Metadata A defining unit of data such as a database table, a data file, a Notes form, or other.

Record An element of data within a table, represented as a row and equivalent to a Notes document.

Result set The set of data resulting from a selection operation: for example, the data to be transferred after being selected through a Notes select statement or SQL statement.

Row An element of data in a table represented as a line of information.

Statement A Notes formula, SQL query or other language string that can be executed against Notes or relational data. A selection statement produces a result set, while other statements modify data.

Table A set of data in a relational database, presented in row and column format.

View A Notes view displays a list of documents with selected fields. The list can be categorized — placing documents in pre-established categories — and hierarchical. In other databases a view is a customized presentation of data from one or more tables.

Terminology Comparison

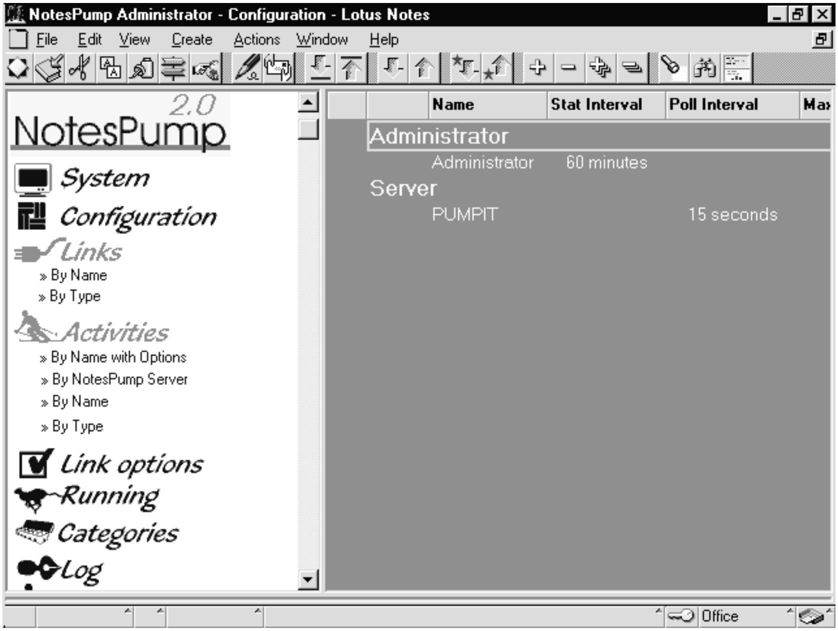
Type	*Relational DB*	*Notes*
Database	Database	Database (.NSF file)
Metadata	Table	Form / View
Field	Column	Field / Column
Record	Row	Document / Row
Statement	SQL query	Selection formula

NotesPump Administration

NotesPump can be managed from any platform supported by Notes, including UNIX, Macintosh, Windows NT, Windows 95 and OS/2. The Notes client is used as the base for system control.

The Administrator database configures and controls NotesPump Servers and their Activities. It contains all the configuration, connection, and scheduling elements that NotesPump uses to collect, transfer, and write data. The information in the database is also known as the Control Store.

The NotesPump navigator for administering the NotesPump Server looks like the following :



Name	Stat Interval	Poll Interval	Max
Administrator			
Administrator	60 minutes		
Server			
PUMPIT		15 seconds	

The elements of the Administrator database (or Control Store) fall into three basic categories that correspond to different aspects of the NotesPump process:

- **Configuration**

  The Administrator database contains configuration information that governs basic aspects of each NotesPump Server that uses that Administrator. A NotesPump Server is one that executes NotesPump Activities. The information applies to all Activities that run on the server.

- **Link**

  The Administrator also holds Link information that defines a connection to the data servers and databases that NotesPump needs to access for transferring data. Link information can be shared by many Activities.

- **Activity**

  The Activity contains instructions for the NotesPump Server and as such it is the central element of NotesPump. Activity information in the Administrator database tells a NotesPump engine what to do, when to do it, and how. NotesPump can execute several kinds of Activities to move and manage data.

Configuration Documents

Two types of Configuration documents are provided:

- Administrator Configuration

  Each NotesPump Administrator database contains a single Administrator Configuration document. It identifies the Notes server on which the NotesPump Log database is installed, the file path of the Log database itself, and the interval in minutes at which each running NotesPump server must notify the Administrator database of its status.

- Server Configuration

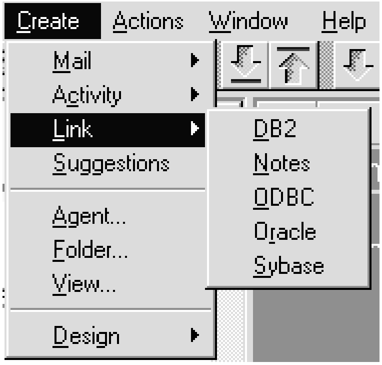
  The Administrator database contains one Server Configuration document for each NotesPump server that it controls. This server document contains the following information:

  - The Current Status of the server
  - The Last Broadcast received from the server
  - The Server Name
  - The Poll Interval — the interval at which the server polls the Administrator for any Activities that should be run
  - The Maximum Number of Activities — the greatest number of concurrently running Activities
  - The Maximum Duration of Activities — the longest duration allowed for an Activity before it is closed by the server
  - The Maximum Consecutive Failures — the number of consecutive failures that an Activity can have before it is disabled by the server

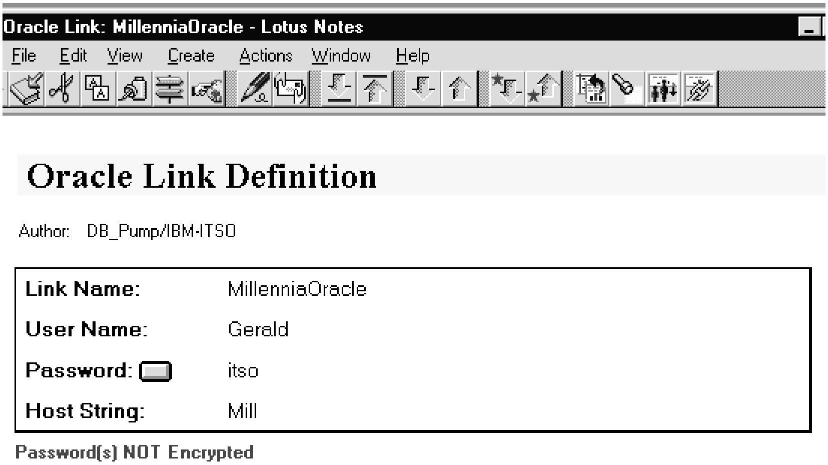
- The Activity Requested Shutdown Time-out — the amount of time allowed for an application to obey a Close command before it is terminated
- International Character Translation — Enabled / Disabled

Link Documents

Link documents are created to establish connectivity to the target of a transfer. When Activities are created, a link to a target database must be chosen from the existing set of links. The Link information will then be copied into the Activity document.



The set of information required in the Link document varies for the different database products. For example, a link with an Oracle database whose instance name is Mill would look like this:



Oracle Link Definition

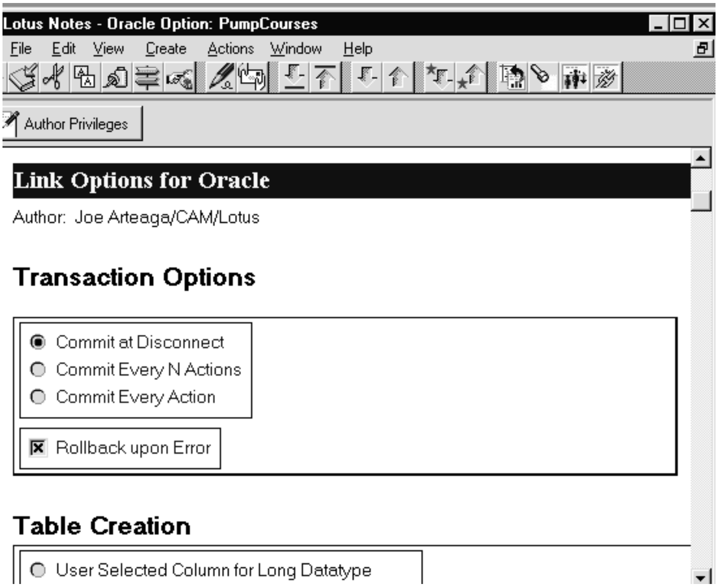
Author: DB_Pump/IBM-ITSO

Link Name:	MillenniaOracle
User Name:	Gerald
Password:	itso
Host String:	Mill

Password(s) NOT Encrypted

Link Options Documents

Link Options documents can be created during the course of building an Activity. They define rules for selecting, manipulating, and writing data by a single Activity, based on the type of system. For example, the Link Options for an Oracle Link Document are shown below:


Lotus Notes - Oracle Option: PumpCourses
File Edit View Create Actions Window Help
Author Privileges

Link Options for Oracle

Author: Joe Arteaga/CAM/Lotus

Transaction Options

● Commit at Disconnect
○ Commit Every N Actions
○ Commit Every Action

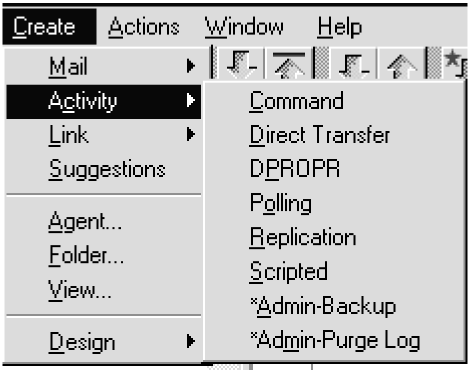
☒ Rollback upon Error

Table Creation

○ User Selected Column for Long Datatype

Activity Documents

Activity documents contain the information that instructs the NotesPump Server to execute an Activity. Each Activity type has its own specific form:



- **Admin-Backup Documents**

  Back up Administrator database data.

- **Admin-Purge Log Documents**

  Purge the NotesPump logs.

- **Command Documents**

  Execute an action against a database.

- **Direct Transfer Document**

  Transfers data from one database to another.

- **DPROPR Document**

  Transfers data from a Data Propagator (DPROPR) Consistent Change Data (CCD) staging table in a relational database to a Notes database.

- **Polling Document**

  Polls a database to see if a specified condition exists, and, if so, executes an Activity.

- **Replication Document**

  Synchronizes data in different databases.

- **Scripted Document**

  Executes an Activity whose data sources and function are defined by a LotusScript script.

While building an Activity, you'll be able to choose Link information from existing Link documents. You should therefore create any Link document that you intend to use before you create the Activity.

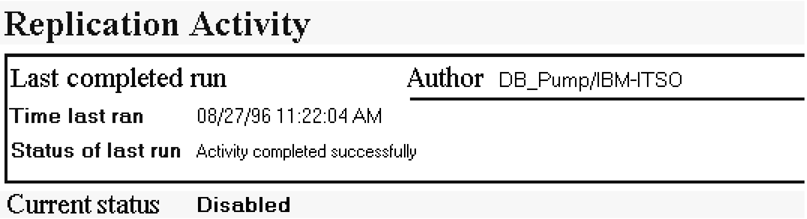
Although each of the Activity documents contains information specific to the type of Activity, the Scheduling section is common to all of them.

Defining Common Areas of Activity Documents

The following areas are common to each of the Activity Documents. Let's look at these areas so that we are familiar with what is in each of them.

Last Completed Run Information

Displays the status of the last run of the Activity.



Replication Activity

Last completed run

Author DB_Pump/IBM-ITSO

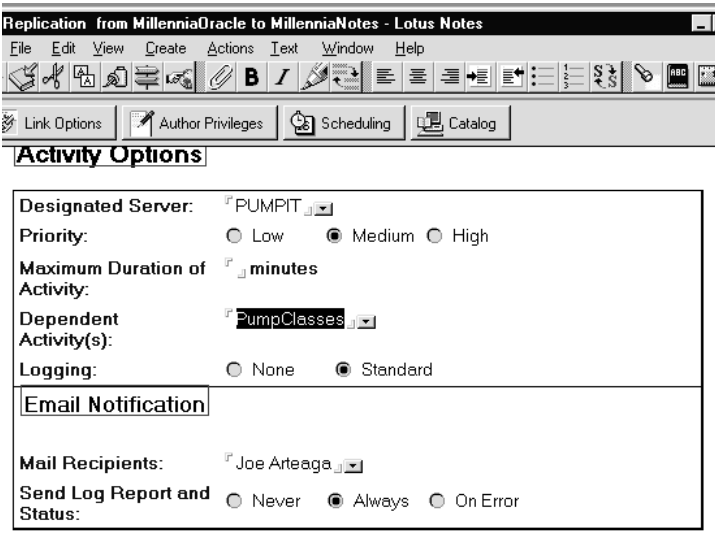
Time last ran 08/27/96 11:22:04 AM

Status of last run Activity completed successfully

Current status Disabled

Activity Options

Define general Activity settings, such as the server to run the Activity on, logging options, any dependent tasks, and priority of the Activity.


Replication from MillenniaOracle to MillenniaNotes - Lotus Notes
File Edit View Create Actions Text Window Help
Link Options
Author Privileges
Scheduling
Catalog

Activity Options

Designated Server: PUMPIT

Priority: ○ Low ● Medium ○ High

Maximum Duration of Activity: minutes

Dependent Activity(s): PumpClasses

Logging: ○ None ● Standard

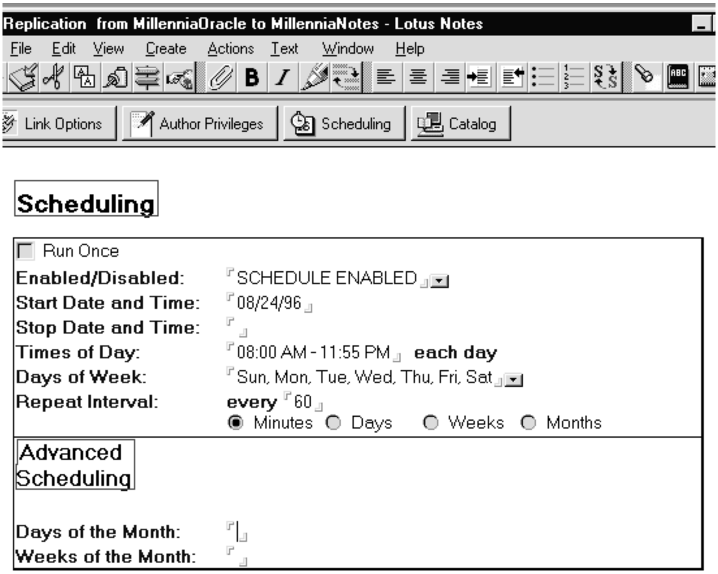
Email Notification

Mail Recipients: Joe Arteaga

Send Log Report and Status: ○ Never ● Always ○ On Error

Scheduling Information

This section defines when and how the Activity will be run. Many different settings are available, to give you flexibility in scheduling. If you want to transfer some information on a one-time basis, check the Run Once check box.


Replication from MillenniaOracle to MillenniaNotes - Lotus Notes
File Edit View Create Actions Text Window Help
Link Options
Author Privileges
Scheduling
Catalog

Scheduling

□ Run Once

Enabled/Disabled: SCHEDULE ENABLED

Start Date and Time: 08/24/96

Stop Date and Time:

Times of Day: 08:00 AM - 11:55 PM each day

Days of Week: Sun, Mon, Tue, Wed, Thu, Fri, Sat

Repeat Interval: every 60
● Minutes ○ Days ○ Weeks ○ Months

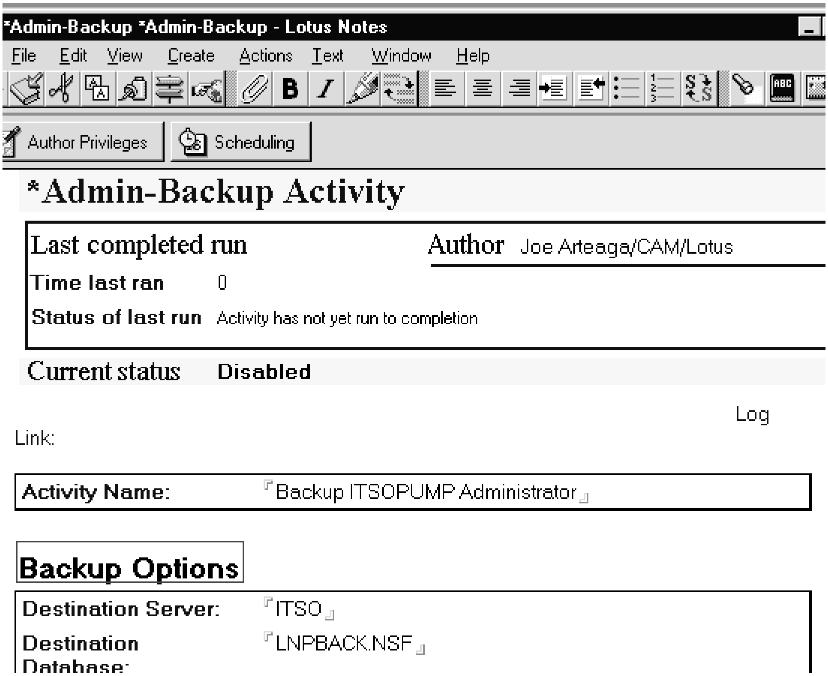
Advanced Scheduling

Days of the Month:

Weeks of the Month:

Admin-Backup Activity Document

An Admin-Backup administration Activity can be used to back up the Administrator database. You can specify a different server and database name for the backup Activity so it can be used if recovery is required.



*Admin-Backup *Admin-Backup - Lotus Notes
File Edit View Create Actions Text Window Help
Author Privileges
Scheduling

*Admin-Backup Activity

Last completed run

Author Joe Arteaga/CAM/Lotus

Time last ran 0

Status of last run Activity has not yet run to completion

Current status Disabled

Log

Link:

Activity Name: Backup ITSOPUMP Administrator

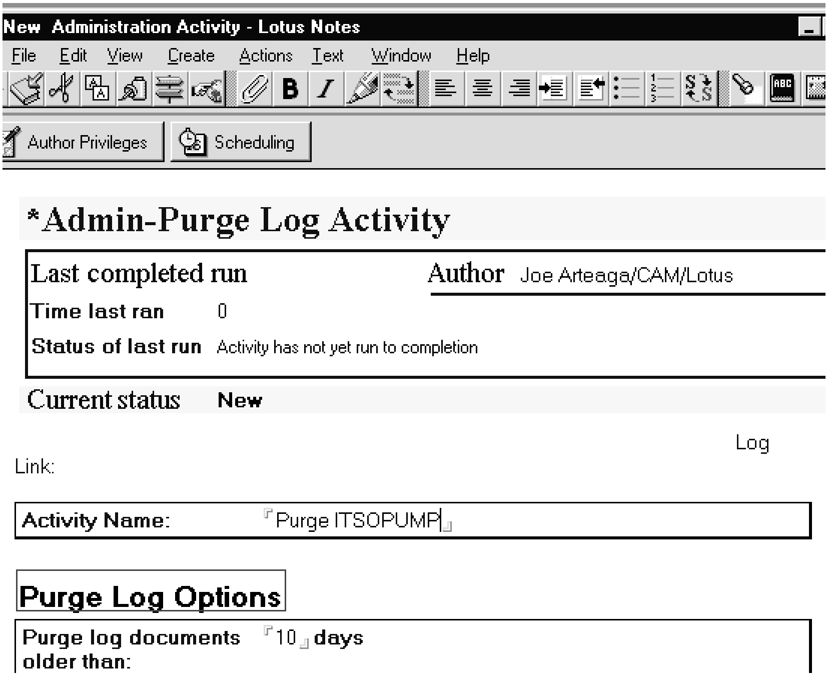
Backup Options

Destination Server: ITSO

Destination Database: LNPBACK.NSF

Admin-Purge Log Activity Document

An Admin-Purge Log Administration Activity can be used to purge the log of documents older than a given number of days. Depending on the logging options you chose in the configuration document and the number of Activities defined, you may need to purge the NotesPump log database. In the example below the database administrator would purge documents older than ten days.



*Admin-Purge Log Activity

Last completed run

Author Joe Arteaga/CAM/Lotus

Time last ran 0

Status of last run Activity has not yet run to completion

Current status New

Log Link:

Activity Name: Purge ITSOPUMP

Purge Log Options

Purge log documents older than: 10 days

Direct Transfer Activity Document

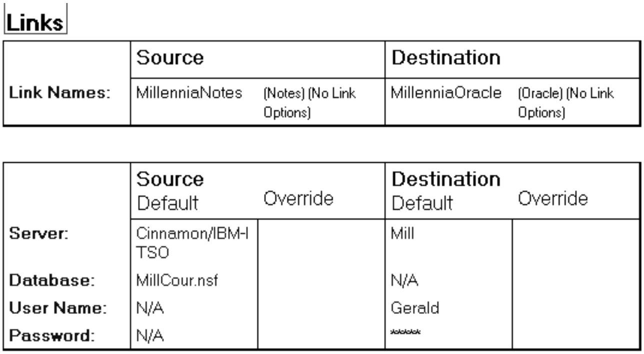
A Direct Transfer Activity transfers data from one database to another. The data to be transferred is identified by a statement, for instance, an SQL query or a Notes selection formula. It can be run once using the Run ASAP button or the task can be scheduled. The Direct Transfer document contains the following information needed for a transfer Activity:

- Links indicating the source and destination databases and metadata
- Command statements for selecting the data to be transferred
- Options controlling the treatment of the data
- Scheduling

Source and destination databases, including access information, must be defined in Link documents before the Direct Transfer Activity is created.

Follow these steps to fill in the Direct Transfer Activity document:

1. Select the Source and Destination Links:

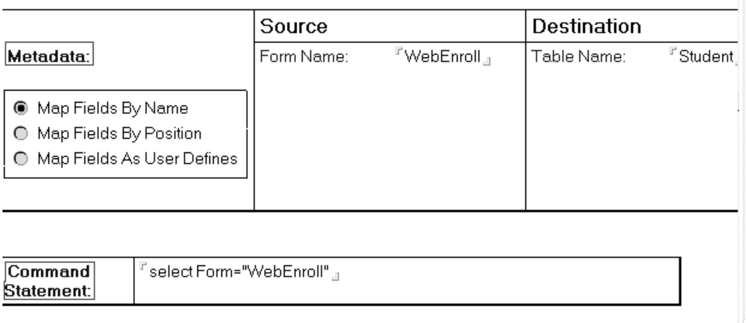


Links

	Source		Destination	
Link Names:	MillenniaNotes	(Notes) (No Link Options)	MillenniaOracle	(Oracle) (No Link Options)

	Source		Destination	
	Default	Override	Default	Override
Server:	Cinnamon/IBM-ITSO		Mill	
Database:	MillCour.nsf		N/A	
User Name:	N/A		Gerald	
Password:	N/A		*****	

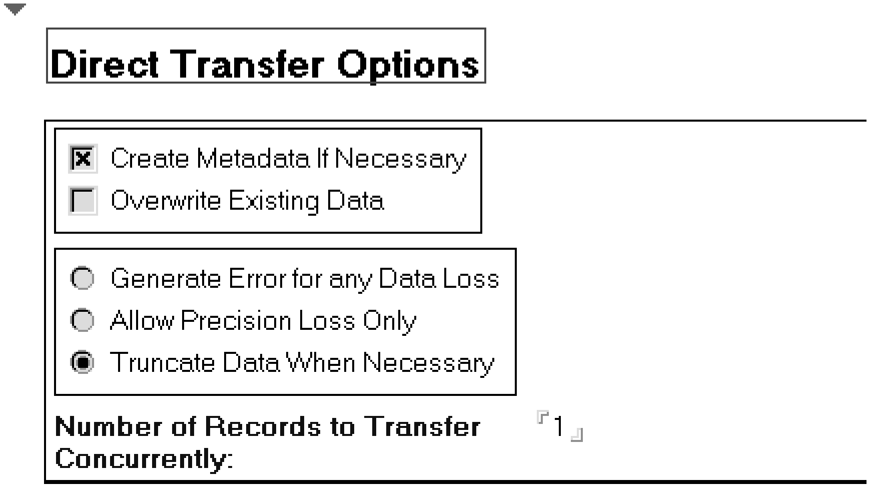
2. Enter the metadata information. For example, the table name in an Oracle Source and the Notes Form in a Notes database, according to the links chosen above:



	Source	Destination
Metadata: ● Map Fields By Name ○ Map Fields By Position ○ Map Fields As User Defines	Form Name: WebEnroll	Table Name: Student

Command Statement:	select Form="WebEnroll"

3. Choose the Direct transfer options. It is important to consider the type of target you will write to. For example, when taking Notes text fields and placing them in Oracle fields, you may want to choose truncation of data as in the example below:



Direct Transfer Options

☒ Create Metadata If Necessary
□ Overwrite Existing Data

○ Generate Error for any Data Loss
○ Allow Precision Loss Only
● Truncate Data When Necessary

Number of Records to Transfer Concurrently: 1

4. Set the Activity and schedule options based on your requirements, and save the Activity.
5. Check your scheduling settings, by looking at the Current Status section in the Document. It will tell you the next time the Activity will run.

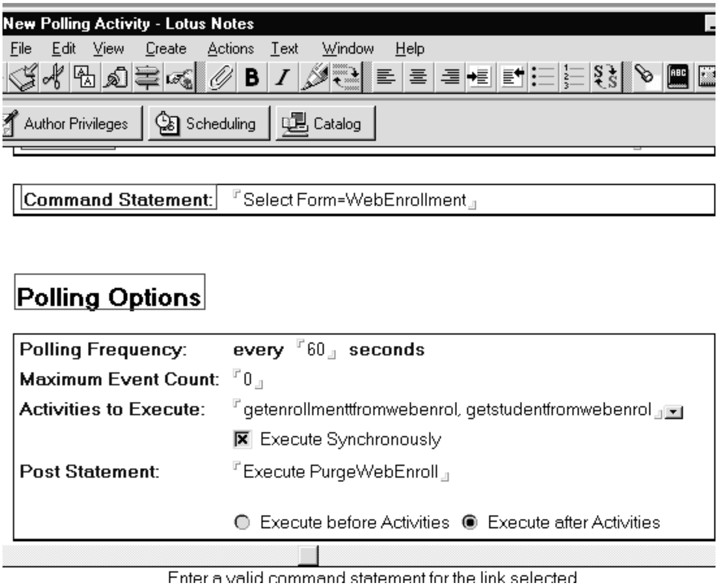
Polling Activity Document

Polling is a way of causing a subordinate Activity (such as a Direct Transfer or Replication) to execute when a condition is met. The Polling document itself does not contain the Activity to execute. Instead, it triggers Activities that are configured in separate Activity documents.

The Polling document contains all information needed to execute subordinate Activities:

- Link — the link to the target
- Condition under which the subordinate Activities — the Activities to Execute — will be run
- Scheduling
- Polling frequency
- Activities to be executed — synchronously or not
- Post statements commands to run before or after the Activities

For example, to check for new enrollments in Notes in the Millennia database, we set up a polling Activity which checked for new documents having been created. If new documents were found, Activities to place the enrollment data into Oracle were run synchronously and then a Notes agent ran on the database to delete those documents.



Command Statement: Select Form=WebEnrollment

Polling Options

Polling Frequency: every 60 seconds

Maximum Event Count: 0

Activities to Execute: getenrollmenttfromwebenrol, getstudentfromwebenrol

☒ Execute Synchronously

Post Statement: Execute PurgeWebEnroll

○ Execute before Activities ● Execute after Activities

Enter a valid command statement for the link selected

Replication Activity Document

Two types of replications are supported:

- Primary Key replication
- Primary Key / Timestamp replication

Primary Key replication

NotesPump primary key replication replicates data based on a key that can comprise one or more specific fields in the data table. The fields must exist in both tables (A and B). The function of the primary key is to determine whether a mismatch of records requires an update to a record, the insertion of a new record, or a record deletion.

The database master identified in the Activity document as the master contains the data that “wins” any replication conflict and from which the other is replicated in Primary Key replication and (under certain conditions) in Primary Key / Timestamp replication.

When the replication takes place, records are updated according to the following rules:

- Record is identical in both databases: no update to the record.
- Record doesn't match but the primary key is identical: the record in the master database updates the one in the non-master database.
- A primary key value exists in the master database but not in the other: The record from the master database is inserted in the other database.
- A primary key value exists in the non-master database but not in the master: The record from the non-master database is deleted.

Note Because no information beyond the primary key is available, all mismatches are regarded as conflicts. Since in the case of conflicts, the master data overwrites the non-master, the master always prevails in a primary-key-only mismatch.

Tip Deletion of records from the non-master database can be prevented by checking the option Do Not Replicate Deletions in the Replication Activity document.

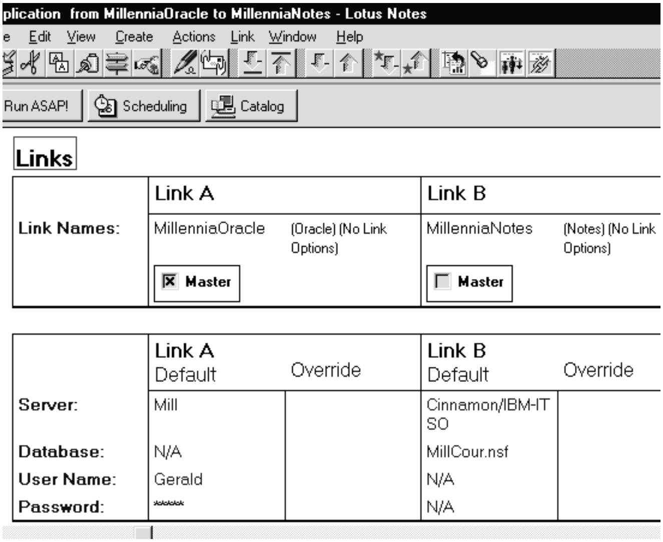
Primary Key/Timestamp Replication

If the database tables involved in the replication contain timestamp fields and these fields are identified in the Activity document, NotesPump performs Primary Key / Timestamp replication. In that case the following process is used to replicate:

1. NotesPump creates two result sets (one for each database) that include only those records that have changed since the last replication, based on the timestamp field.
2. NotesPump then matches primary keys, and:
   - If both master and non-master records are found, the master record updates the other.
   - If only one record is found (either master or non-master), NotesPump attempts to locate the primary key in the other database. If the other key is found, the changed record updates the one that was not changed. If the other key is not found, a copy of the changed record is inserted into the other database.

Note Because NotesPump looks only at changed records, deletions are not replicated if a record is deleted from one database and the matching record in the other database remains unchanged.

For example, to replicate courses existing in an Oracle database to a Notes database, the primary components of the document are shown next.

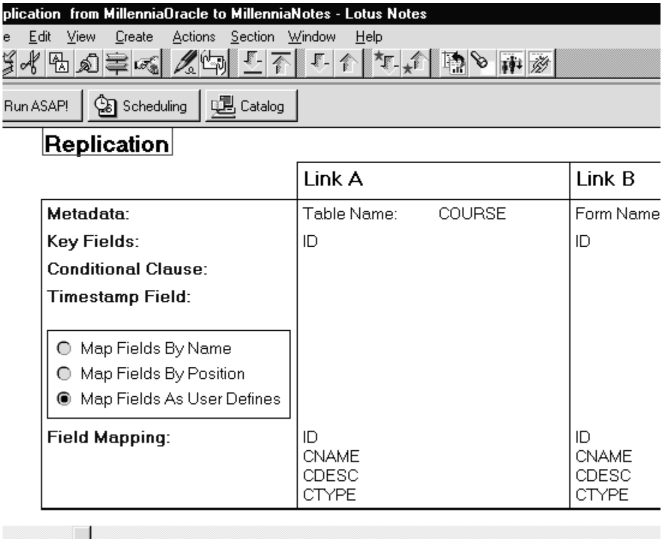

plication from MillenniaOracle to MillenniaNotes - Lotus Notes
e Edit View Create Actions Link Window Help
Run ASAP!
Scheduling
Catalog

Links

	Link A		Link B	
Link Names:	MillenniaOracle ☒ **Master**	(Oracle) (No Link Options)	MillenniaNotes □ **Master**	(Notes) (No Link Options)

	Link A Default	Override	Link B Default	Override
Server:	Mill		Cinnamon/IBM-ITSO	
Database:	N/A		MillCour.nsf	
User Name:	Gerald		N/A	
Password:	*****		N/A	

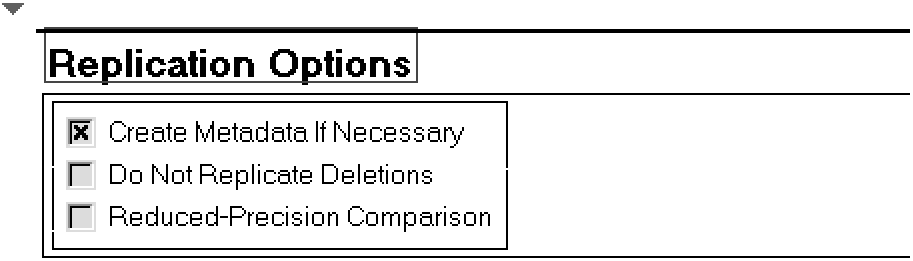
The Replication settings for Source and Destination Links are then set up. Notice the fields are explicitly defined.


plication from MillenniaOracle to MillenniaNotes - Lotus Notes
e Edit View Create Actions Section Window Help
Run ASAP!
Scheduling
Catalog

Replication

	Link A	Link B
Metadata:	Table Name: COURSE	Form Name
Key Fields:	ID	ID
Conditional Clause:		
Timestamp Field:		
○ Map Fields By Name ○ Map Fields By Position ● Map Fields As User Defines		
Field Mapping:	ID CNAME CDESC CTYPE	ID CNAME CDESC CTYPE

The Replication Options will create a new Notes form if needed. Since we want the Oracle database to always be the master, the Do Not Replicate Deletions check box is not checked. This way any documents with a key not in Oracle will be deleted, and the two systems will be synchronized. We could also set the Reduced Precision option to avoid false mismatches based on differences in the way a particular DBMS stores floating point, timestamp, and datetime values, as opposed to another DBMS system or Notes.



Replication Options

☒ Create Metadata If Necessary
□ Do Not Replicate Deletions
□ Reduced-Precision Comparison

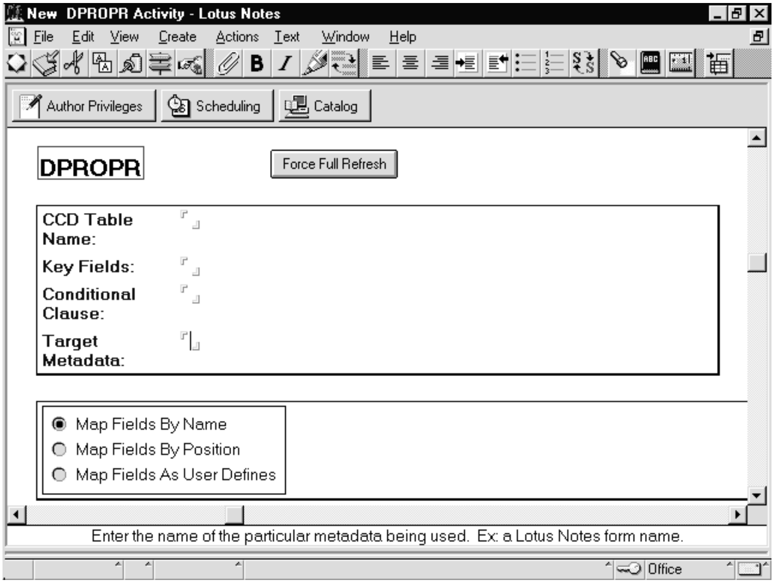
DPROPR Activity Document

A DPROPR Activity transfers data from an IBM Data Propagator (DPROPR) Consistent Change Data (CCD) staging table in a relational database to any NotesPump data source. The data to be transferred consists of changes to IBM DB2 tables.

The DPROPR Activity document contains information needed for a DPROPR transfer Activity:

- Links indicating the source and target database and metadata
- Options controlling the treatment of the data, in normal and error recovery situations
- Scheduling

Source and target databases, including access information, must be defined in Link documents before the DPROPR Activity is created.


New DPROPR Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Author Privileges
Scheduling
Catalog
DPROPR
Force Full Refresh

CCD Table Name:

Key Fields:

Conditional Clause:

Target Metadata:

● Map Fields By Name
○ Map Fields By Position
○ Map Fields As User Defines

Enter the name of the particular metadata being used. Ex: a Lotus Notes form name.

CCD table name The CCD table is a DB2 relational table — specifically a condensed, complete, Consistent Change Data table created by a DPROPR subscription or another mechanism.

Key Fields The fields or columns of the data that together represent a unique primary key for that data collection. No two records can have the same value for all key fields, and at least one key field is required. The target key field will be defined to be the same as the source key field.

Conditional Clause Used to further refine the Replication Activity. May be used to specify additional clauses to be used in the selection of rows from the CCD table, for example, Dept=7.

Target Metadata Specifies a Notes form or RDB table where data will be stored.

Force Full Refresh Resets the Sequence ID used by DPROPR to determine what changes to replicate. Resetting this ID and running the Activity causes a full refresh of the target Notes metadata.

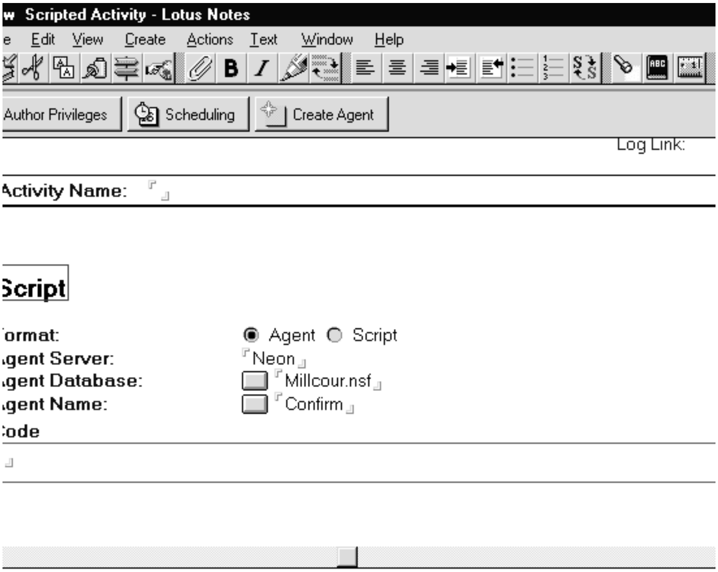
Note NotesPump does not transfer the CCD internal columns IBMSNAP_INTENTSEQ, IBMSNAP_OPERATION, IBMSNAP_COMMITSEQ, and IBMSNAP_LOGMARKER unless you specify them in the field mapping.

For more information about IBM DataPropagator Relational, search on DataPropagator Relational from the IBM home page: http://www.software.ibm.com. Or dial 1-800-IBM-3333 for more information on IBM products and services.

Scripted Activity Documents

If you want greater control of the NotesPump data transfer process, you can create a programmable Activity to interact with the data. NotesPump provides the Scripted Activity for this purpose.

A sample scripted Activity document follows:



w Scripted Activity - Lotus Notes
e Edit View Create Actions Text Window Help
Author Privileges
Scheduling
Create Agent

Log Link:

Activity Name:

Script

ormat: ● Agent ○ Script
gent Server: Neon
gent Database: Millcour.nsf
gent Name: Confirm
ode

The Script section of the document identifies the LotusScript code that the Activity will use. By identifying the Agent server and database, you tell NotesPump where the agent containing the script can be found and run. If your script is small enough, click on the Script check box and enter the code in the Code field.

The Scripted Activity document also has the scheduling and server tuning features found in the other Activities. However, since the processing of data is expected to be done through LotusScript, the Activity does not have a section for links and metadata processing.

Through the NotesPump extensions to the LotusScript language, data may be queried, modified, normalized, etc. These extensions provide access to servers, databases, tables, records, fields, and other metadata.

In a simple transfer of data from one database connection to another, the script connects to the two points, selects the data of interest, and then fetches from one and inserts into the other. More complicated actions such as data massaging, replication conflict resolution and multi-point data manipulation are all possible.

A NotesPump scripted Activity is developed with the following programming flow:

- Initialize Activity
- Create Connection to Data
- Connect to Database
- Select Data of Interest
- Fetch Data
- Manipulate Data
- Insert Data
- Disconnect

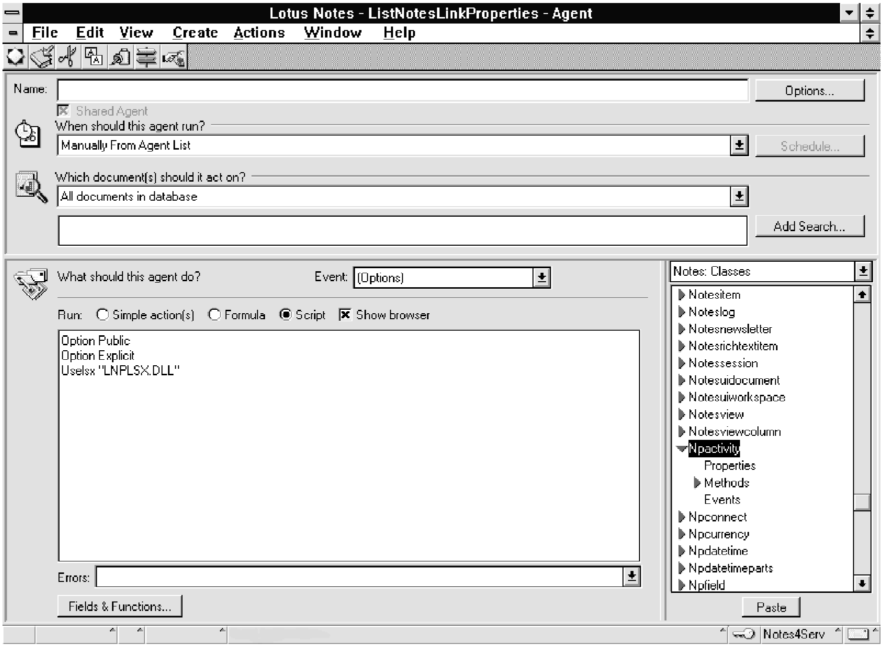
Using NotesPump Extensions

To use the NotesPump extensions to LotusScript, include the NotesPump LSX within a scripted agent. This is accomplished with the UseLSX statement. The NotesPump installation registers its LSX so that the following platform independent syntax may be used within a scripted agent's "options" section:0

```
UseLSX "*lnplsx"
```

For your agent to execute, Notes and NotesPump need their program directories on the system path. Notes requires the NotesPump program directory to be on the path so the IDE may load the LSX for authoring and testing of the agent. NotesPump requires the Notes program directory to be on the system path so that the agent may be executed as part of the scripted Activity.

The following figure shows the effect of using the UseLSX command above. Notice the NotesPump classes available to the LotusScript agent in the IDE browser pane.



Name:

☒ Shared Agent

When should this agent run?
Manually From Agent List

Which document(s) should it act on?
All documents in database

What should this agent do? Event: (Options)

Run: ○ Simple action(s) ○ Formula ● Script ☒ Show browser

```
Option Public
Option Explicit
Uselsx "LNPLSX.DLL"
```

Errors:

Notes: Classes

- Notesitem
- Noteslog
- Notesnewsletter
- Notesrichtextitem
- Notessession
- Notesuidocument
- Notesuiworkspace
- Notesview
- Notesviewcolumn
- Npactivity
  - Properties
  - Methods
  - Events
- Npconnect
- Npcurrency
- Npdatetime
- Npdatetimeparts
- Npfield

You can test the agent through the Lotus Notes IDE by turning the Debug LotusScript option in the File - Tools menu.

Tip When you create the agent, make it shared with these settings:

- Select either “manually from the Actions Menu” or the “manually from the Agent List.”
- Select Run Once.

LotusScript Extensions

The following are the classes available for your agent’s LotusScript code when you load the LotusScript NotesPump Extensions. Please refer to the book “NotesPump Extensions to the LotusScript Language” for a more detailed list of the class methods and properties. An example of the use of these classes is presented following the class descriptions.

NP Activity class

Each Activity has global state information which is managed by NotesPump. This state is used to manage Link libraries and connections, allocated objects, error and event information, count logs, and communications with the NotesPump Server.

NPConnect class

Connect functions manage individual Link connection contexts within an Activity. The Connect class enforces Link state and requirements, and must always be used when interacting with Links. One connect object should exist for each individual data connection through a Link used in an Activity. The Activity context includes error handling, which is also available to Links. Error information may be added by the Link or the calling Activity with the NPActivityLog functions. Connect functions automatically update information on counts of records affected and functions called in the Activity context. This information is logged at Activity termination.

NPCurrency class

A currency value is represented by object NPCURRENCY. The value is an 8-byte integer with a fixed scale of 4, providing 19 digits of precision. Currency is commonly used when complete precision is required, such as for monetary amounts. A currency is much more precise than an integer, more precise than a float, and more efficient than a numeric. Note that during any currency overflow, the maximum or minimum valid currency value is assigned in addition to the error generated.

NPDatetime class

A datetime value is represented by the object NPDATETIME. A datetime value represents a specific date and time, including timezone and daylight savings time information. A datetime value may have either its date or time component unavailable, indicated by special constant values for date (NPDTNULL_DATE) or time (NPDTNULL_TIME) components. During any datetime overflow, the maximum or minimum valid datetime value is assigned in addition to the error return.

NPField class

A field is a data object containing one or more data values of a designated data type. A field is represented by the handle type NPFIELD.

A field instance contains the following information:

- Data Type

  One of the NotesPump datatypes. Assigned at creation and cannot be changed.

- Value Count

  One or greater. Data space for the value specified and NULL indicators will be allocated. The value count is automatically assigned for fields in a FieldList based on the FieldList record count (see Fieldlist description). Assigned at creation and cannot be changed.

- Field Flags

  Zero or more field flags OR-ed together. See Field Flags description below. May be changed at any time.

- Format

  Data-type specific data source format information. See Field Format description below. May be changed at any time, although changes to the format clear all data values for the field.

- Virtual Code

  Link or Connection virtual code. May be changed at any time.

- Data

  For each value of this field (see Value Count), a single data value and NULL indicator are allocated. These values are stored separately, but for a multi-value field, the values and indicators for consecutive field value indices are stored in consecutive memory.

NPFieldlist class

A fieldlist represents metadata for a record and may contain data for one or more record values. A fieldlist is a list of fields and field names with supporting functions. Fields within a fieldlist can be added, modified, retrieved, or listed in multiple ways. A fieldlist is represented by the handle type NPFIELDLIST.

A fieldlist instance contains the following information:

- Sequence Number

  A number which is unique across all fieldlists and all variations of a single fieldlist. See Fieldlist Sequence Number description below. Dynamically altered by the API.

- Record Count

  Zero or greater. Data space for this many data values per field will be allocated. The record count is assigned as the value count for all fields added to this fieldlist. A value of zero indicates a fieldlist which contains no data, but is used solely for field name tracking and fieldlist merging. See Fieldlist Merging description below. Assigned at creation and cannot be changed.

- Default Field Flags

  Default NPFIELDF_XXX flags which are assigned to all new fields added to the fieldlist. This is commonly NPFIELDF_TRUNC_PREC to allow precision loss only. Assigned at creation and cannot be changed.

- Field Count

  Number of fields currently in the fieldlist. This changes to reflect all fields added to or removed from the fieldlist.

- Field Array

  List of fields in the fieldlist. All fields allocated by the fieldlist are contained in this list. These fields are owned by the fieldlist and are freed with the fieldlist, not with NPField Destructor. Fields are added to a fieldlist with NPFieldlistAppend and NPFieldlistInsert; replaced with NPFieldlistReplace; and removed with NPFieldlistRemove.

- Field Name Arrays

  These lists contain the field names for each field in different stream formats. For efficiency, field names may be cached in multiple stream formats, when requests for field names from a fieldlist occur in multiple stream formats. For example, if a fieldlist is sent to two Links, one using IBM Code Page 932 and the other using the LMBCS character set, the fieldlist retains both formats of field names.

NPFloat class

A float value is a standard IEEE 8-byte floating-point value. Normal manipulation of these values should be performed using native language functionality.

The only additional functions supported by the NP API convert float values to and from currency, numeric, and stream data types.

NPInt class

An integer value is a standard IEEE 4-byte signed integer. Normal manipulation of these values should be performed using native language functionality. The only additional functionality supported by the NP API is conversion to and from streams.

NPNumeric class

A numeric value is represented by the object NPNUMERIC, containing a precision, scale, and variable number of digits. Numeric values have a specific precision and scale, and can accommodate high-precision numbers. Before being used, a numeric value must either be zeroed or created using NPNumeric.Create. A zeroed numeric submitted to any other numeric function is initialized to a precision of 88 and a scale of 44. Note that during any numeric overflow, the maximum or minimum valid numeric value is assigned in addition to the error return.

For a numeric to be valid, it must have valid values for precision and scale settings. A numeric can be initialized with NPNumericCreate. If precision and scale are not valid for a numeric submitted as a parameter to a Numeric function, then an error may be generated. If the invalid numeric has been zeroed, then it will be automatically initialized to a numeric with the maximum precision and a scale of precision/2 (44). If the invalid numeric has not been zeroed, then an NPFAIL_INVALID_NUMERIC error will be returned.

NPStream class

A stream represents two of the seven NotesPump datatypes, text and binary. A stream value is a variable length list of characters or bytes. Streams come in two basic types, text and binary, represented by the corresponding NotesPump data types. Specific format information indicates either the character set (for text) or the binary format (for binary). No NPStreamData functions should be called until the stream has been zeroed, NPStreamClear has been called, or NPStreamCreate has been called.

Example: Scripted Activity

What follows is a simple example showing how LotusScript may be used to handle processing for NotesPump Scripted Activities.

In this example, we need to bill US travelers for purchases from around the world. Each record contains a product description, the country of origin, and a cost in the country's currency. We wish to bill these items; however, we must bill them in US dollars.

We will first locate the up-to-date exchange rates for the currencies we will process. Next we will use the direct transfer script. Remember, a direct transfer simply connects to source and target databases, selects a set of records from the source, then fetches those records from the source and inserts them into the destination.

We add a little data processing to the transfer. After we fetch the record from the source database, we check the country of origin, then, using the corresponding exchange rates, we convert the product cost to US dollars, and insert the record into the target database. As an added feature, we update the source record to indicate that the transaction has been processed.

Scripted Activity Code Example

```
'**(Options)
Option Explicit
'** UseLSX command loads NotesPump extensions to LotusScript
Uselsx "*lnplsx"

'**(Forward)
Declare Sub Initialize
Declare Sub GetExchangeRates (Rates As NPField)
Declare Sub BillTransactions (Rates As NPField)
```

```
'**(Declarations)
Dim SectionName As String '** we use this to track the
section of script which is active, this improves our error
handling / reporting
Dim State As NPActivity

'** the following constants must be changed to correspond to
real Activities, links and database tables
Public Const ACTIVITY_NAME = "Name of This Activity Goes
Here"
Public Const RATES_LINK = "Name of Exchange Rates Link Goes
Here"
Public Const RATES_METADATA ="Name of Exchange Rates Table
Goes Here"

Public Const SOURCE_LINK ="Name of Source Link Goes Here"

Public Const DESTINATION_LINK ="Name of Destination Link Goes
Here"

Public Const SOURCE_METADATA ="Name of Source Table Goes
Here"

Public Const DESTINATION_METADATA ="Name of Destination Table
Goes Here"

'** The Initialize subroutine is the main body of the script.
Sub Initialize

'** the very first code in a LotusScript program should be an
error handler. This will help to develop and test the script.
The error handler for this script is located at the end of
the Initialize subroutine and is available to the rest of the
module to aid in the tracking of errors, we also use a global
string to track the section which is currently active. In
case of an error, the error handler uses this string to
indicate more exactly where the problem occurred
           On Error Goto LSErrorLabel
           SectionName = "Initialize"

'** Before any NotesPump objects may be used, an NPActivity
context must be created. This provides all of the logging and
administration required by the rest of NotesPump and the
other NotesPump objects you may create.

           Set State = New NPActivity (0, ACTIVITY_NAME)
```

```
'** Now we get down to business ...
'** We locate all of the exchange rates needed to process the
'** received transaction so that they may be billed in US
'** dollars. Connect to the exchange rate database

           Dim Rates As New NPField (NPTYPE_FLOAT, 6)
           GetExchangeRates Rates

'** we locate all of the transactions which have been
received and must now be billed.

           BillTransactions Rates
'**we have completed without an error, so we are finished

           Goto ResumeLabel

LSErrorLabel:
           Dim MessageString As String
           MessageString = "Error at line " & Cstr(Erl()) & "
in section " & SectionName

           If (State Is Nothing) Then
                        Print MessageString
           Else
                        Dim ErrorStream As New NPStream
                        ErrorStream.AsLsString =
MessageString
                        State.LogStream 0, ErrorStream, Err()
           End If

           Resume ResumeLabel

ResumeLabel:
'** we do not need to explicitly delete any of the objects we
have created. LotusScript will handle clean up as objects go
out of scope. Done

End Sub

'** GetExchangeRates subroutine opens an Oracle table which
contains the most recent currency exchange rates for a set of
countries. These rates are store in an array to be used later
to convert foreign currency totals to US dollars

Sub GetExchangeRates (Rates As NPField)
           Dim SavedSectionName As String
           SavedSectionName = SectionName
           SectionName = "GetExchangeRates"
```

```
'** create a link to the Oracle database
Dim RateSource As New NPConnect (0, RATES_LINK

Dim KeyName As New NPStream
Dim Keys As New NPFieldList (1,0)
Dim SearchKey As Variant
Dim Record As NPFieldlist
Dim Count As Long
Dim Index As Long

'** using the expanded properties of the Oracle
'** connection, we specify which table contains
'** the records of interest
RateSource.Metadata = RATES_

'** we now connect to the Oracle database and
'** table

RateSource.Connect

'** Here is an example of using a selection key
'** for locating records.
'** We create a lookup key list to locate each
'** exchange rate. The search key list is built
'** and then the value to search for is assigned
'** inside the loop below

KeyName.AsLSString = "REGIONID"
Keys.Append KeyName, NPTYPE_INT, SearchKey
SearchKey.SetFlags (NPFIELDF_KEY)

For Index = 1 To 6
             Set Record = New NPFieldList (1, 0)

             '** set up the key for our search
             Rates.ValueIndex = Index
             SearchKey.AsLSLong = Index

             '** now that we have defined the
             '** search key, we locate the
             '** corresponding record
             RateSource.Select Keys, 1, Record,
             Count
             If ((Count = 0) Or ((Count =
             NPCOUNT_UNKNOWN) And (Record.GetCount
             = 0))) Then Goto CleanUp
             '** now we fetch the record
             RateSource.Fetch Record, 1, 1, Count
```

```
           '** Rate will be stored internally as NPFLOAT
           '** since the field was defined that way
           Rates.AsLSString = Record.ExchangeRate

                        '** reset record
                        Delete Record
           Next
           CleanUp:
           SectionName = SavedSectionName
End Sub

'** the BillTransactions subroutine opens an Oracle table
which contains the unprocessed transactions. Using the
exchange rates, extracted previously, the orders are
converted to US dollars and then stored in the target table

Sub BillTransactions (Rates As NPField)
           Dim SavedSectionName As String
           SavedSectionName = SectionName
           SectionName = "BillTransactions"

           '** create two links to the Oracle database we
           '** will differentiate them below by indicating
           '** different tables

           Dim Source As New NPConnect (0, SOURCE_LINK)
           Dim Destination As New
           NPConnect(0,DESTINATION_LINK)

           Dim KeyName As New NPStream
           Dim Keys As New NPFieldList (1, 0)
           Dim SearchKey As Variant
           Dim Record As New NPFieldlist
           (1, NPFIELDF_TRUNC_PREC)

           Dim Total As New NPField (NPTYPE_FLOAT, 1)
           Dim TState As New NPField (NPTYPE_TEXT, 1)
           Dim RegionID As New NPField (NPTYPE_INT, 1)

           Dim Count As Long
           Dim Index As Long

           '** indicate which table contains the unprocessed
           '** transactions
           Source.Metadata = SOURCE_METADATA
```

```
'** we write-back updates to the source indicating the
           '** transaction has been processed
           Source.Writeback = "1"
           '** connect to the transactions databases
           Source.Connect

           '** indicate which table receives the completed
           '**transactions
           Destination.Metadata = DESTINATION_METADATA

           '** connect to the billing databases
           Destination.Connect

           '** this is an example of executing a database
           '** specific selection formula.
           '** In this example it is an Oracle SQL statement

           Dim Statement As New NPStream
           Statement.AsLsString = "Select * FROM
           INTERNATIONALSALES WHERE TSTATE = 'Entered'"
           Source.Execute Statement, Record, Count

           '** this is an example of using a selection key
           KeyName.AsLSString = "TState"
           Keys.Append KeyName, NPTYPE_TEXT, SearchKey
           SearchKey.SetFlags NPFIELDF_KEY
           SearchKey.AsLsString = "Entered"

           Source.Select Keys, 1, Record, Count

           If ((Count = 0) Or ((Count = NPCOUNT_UNKNOWN) And
           (Record.GetCount = 0))) Then Goto CleanUp

           '** We wish to perform data massaging, so we

           '** locate the field(s) of interest from the x
           '** loop. Since the field positions within the
           '** do not change, fetched data from a link within
           '** the fieldlist loop will be available in the
           '** fields you lookup here. This eliminates the
           '** loop.lookup process within the

           Record.Lookup "Total", Total, 0
           Record.Lookup "TState", TState, 0
           Record.Lookup "RegionID", RegionID, 0

           '** Loop through load-store sequence
           Source.Fetch Record, 1, 1, Count
           If (Count = 0) Goto CleanUp
```

```
'** We don't care if the table exists in these
'** nextlines so we will start to ignore NotesPump
State.RaiseExceptions = False
errors
'** Un-comment the following lines if you wish to
'** the target table
'** Destination.Create NPOBJECT_METADATA, Record
'** if the table has already been created, don't
'** of it as a fatal error
'** thinkState.ClearStatus

'** Un-comment the following lines if you wish to
'** truncate the target table
'** Destination.Action NPACTION_TRUNCATE
'** State.ClearStatus

'** Now restore the automatic handling of
'** errors
State.RaiseExceptions = True


'** Now we loop through each unprocessed
transaction

While (Count > 0)
'** We wish to perform data massaging, so we now
'** effect the fields we located above. First we
'** mark the new transaction as processed so we
'** don't bill it twice

Tstate.AsLSString = "Processed"
Source.Update Record, 1, 1, Count

'** Now we locate the correct exchange rate for
'** this transaction's origin, and convert the
'**  total

Rates.ValueIndex = RegionID.AsLSLong
Total.AsLSDouble=Total.AsLSDouble /
```

```
          '** Rates.AsLSDouble now we store the data in the
          '** target table
          Destination.Insert Record, 1, 1, Count

          '** Finally, we fetch another record and repeat
          '** the loop

          Source.Fetch Record, 1, 1, Count
          Wend

CleanUp:
          SectionName = SavedSectionName
End Sub
```

Activity Field Matching

During a Direct Transfer, Replication or DPROPR Activity NotesPump maps columns in the source and target databases in one of three ways: by position, by field name or by selective mapping.

- According to field position

  Data in the first field of the source result set is copied into the first field of the target database, and so on. If the command statement in the Activity selects a subset of columns from the database table, they will be treated as a simple incremental series starting with 1. The order in which the columns are listed in the selection statement is the order in which they will be transferred. The SQL statement "SELECT column6, column4, column7 FROM table" maps column 6 of the source to field 1 of the target, column 4 to field 2, and column 7 to field 3.

- According to field names

  Data is transferred from a field in the source database to a field of the same name in the target. If the Activity's command statement is an SQL query, you can use column aliases to match the source field to the destination: for example "SELECT empno employee" maps the empno field in the source to the employee field in the target.

  Note The actual expression depends on the database SQL syntax.

- According to selective mapping

  Field names are provided by the user for both the source and target, and are mapped in the order provided. For example, source fields listed as "A, B, C" and target fields listed as "X, Y, Z" result in source field A moving to target field X, B to Y, and C to Z.

When Field Numbers Don't Match

Since Replication requires the same number of columns in both databases, mismatching numbers of columns only affect Direct Transfer and DPROPR.

- More columns in the source result set than in the target metadata

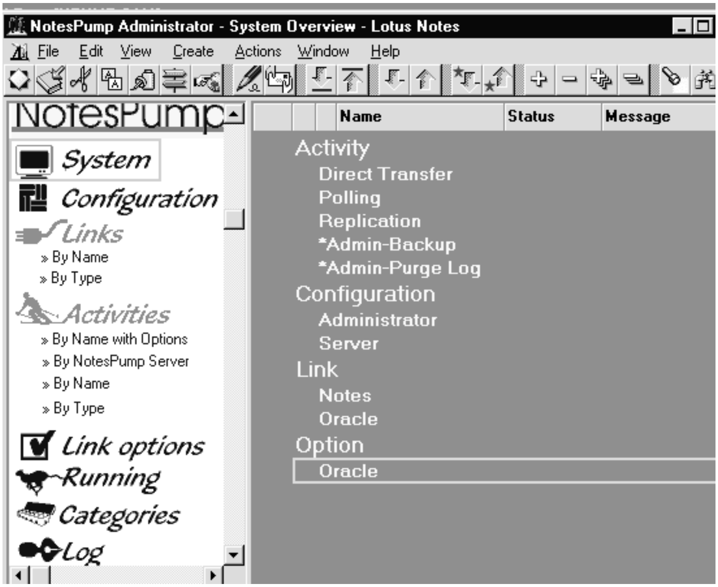
  The transfer will fail and you'll get an error message.

- Fewer columns in the source result set than in the target metadata set

  The transfer will succeed and the "orphan" columns in the target will be filled with null values. If the target database cannot accommodate nulls, the transfer will fail and you will get an error.

Administrator Views

The Administrator is provided with different views of the Administrator database in order to ease setup, controlling, and status checking. The following views are available:



Name | Status | Message

Activity
  Direct Transfer
  Polling
  Replication
  *Admin-Backup
  *Admin-Purge Log
Configuration
  Administrator
  Server
Link
  Notes
  Oracle
Option
  Oracle

Log Views and Documents

The NotesPump Log records all events. The logged document types are: Activity, operation, and server.

Each log document records a separate event, such as an Activity from start to end.

Log Views

Log documents can be listed through this view:

- **All by Server** — Lists all Activities categorized by server. Under the server name, documents are categorized by start date.
- **All by Type** — Lists all Activities categorized by type (Activity, operation, and server). Documents are further categorized by server and start date.

Log Documents

There are three types of Log Documents. They record Activities, Operation events, and Server events.

Activity

The Activity log document lists event and status information associated with an Administration, Direct Transfer, Polling, or Replication Activity.

Operation

The Operation log document lists errors that occurred in the NotesPump application.

Server

The Server log document lists server events, such as startup, termination, Activity execution, and any server termination error conditions.

NotesPump Agents

Three agents are supplied with NotesPump to allow for manipulating Administration database documents by either the Administrator or the NotesPump System.

RunASAP

RunASAP causes a selected Activity to be run as soon as possible. It is the equivalent of the RunASAP button in the Activity document.

ClearLock

The ClearLock agent clears an Activity document's Lock field and related fields so that the Activity can be run again.

DeleteOrphanOptions

DeleteOrphanOptions marks for deletion those Link Options documents that are orphaned because the Activity documents with which they are associated were deleted.

About Scheduling

Data transfer Activities can be scheduled for repetitive days, times, or selections of times. Multiple Activities can be scheduled to execute as dependent processes. In addition, advanced scheduling options can be constructed to allow the execution of an Activity based on time-sensitive business conditions. An example of this flexibility would allow NotesPump to run an Activity on the last five business days of the month.

Running an Activity From the Command Line

NotesPump allows Activities to be executed from the system command line of a running NotesPump server. This feature allows programs with the ability to execute system commands, to start NotesPump Activities. The syntax for starting a NotesPump Activity from the command line is as follows:

LNPACT "*ActivityName*"

The NotesPump server only runs Activities which have been scheduled from the Administrator (including those which have been marked Run ASAP). Also, when an Activity is running from a NotesPump server, it will not run simultaneously on that or any other server until it has completed its current execution. This is referred to as “exclusive” execution. In contrast, when an Activity is started from the system command line, its execution may be “shared.”

This means that the Activity may be run more than once simultaneously. At first this may not seem very safe or useful but there are special cases where this is important; performing searches for an Internet Web application or generating reports are two examples where the “shared” execution is valuable.

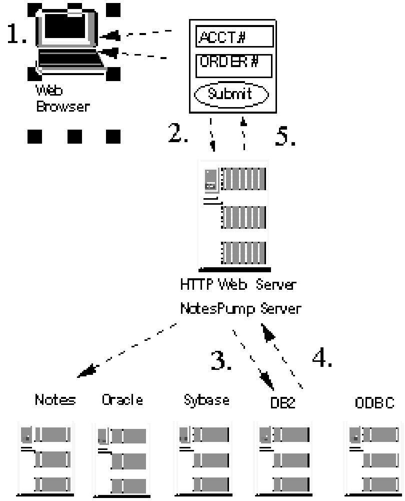
Common Gateway Interface for NotesPump

NotesPump ships with a Common Gateway Interface which allows it to be integrated into enterprise Internet/Intranet Hypertext Transfer Protocol (HTTP) servers for Web applications. Using the NotesPump CGI provides Web clients controlled access to any NotesPump supported database. The interface acts much like the system command line interface, LNPACT. When the CGI is launched via a web server, all of the fields of corresponding HTML form become extended properties of the Activity (see the NPActivity class description in the NotesPump LSX documentation for an explanation of extended properties). After creating an HTML form, simply use the syntax for the POST command as follows:

```
<FORM METHOD="POST" ACTION="ACTCGI.EXE?ActivityName">
```

A diagram showing the flow of such an Activity is shown below.

CGI and NotesPump for Web Client Data Access



1. Customer enters "Acct.#", "Order #" to HTML Web browser form

2. NotesPump launched via HTTP CGI-BIN program, initiates query request "OrderStatus".

3. "OrderStatus" NotesPump Activity initiates on NotesPun Server, obtains Activity name definition from NPAdmin

4. NotesPump processes "OrderStatus"; queries database and encapsulates results to HTML output

5. HTTP Server returns HTM output to Web browser requester

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:

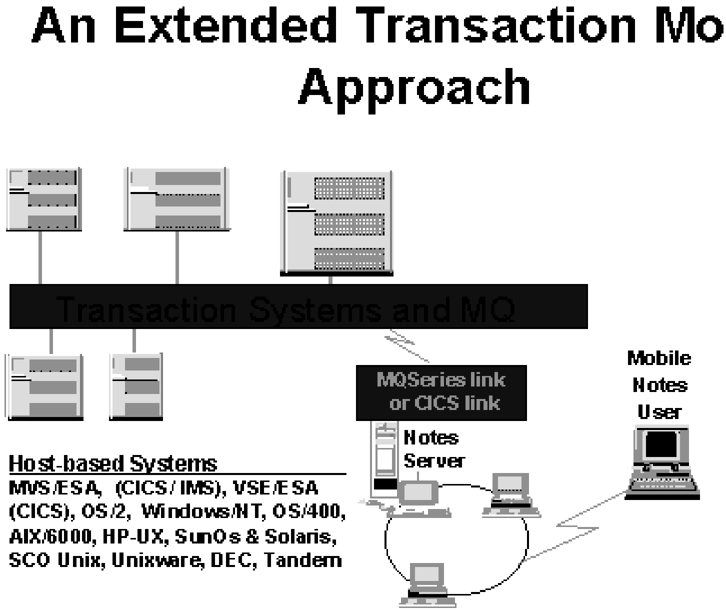
   IBM MQSeries™ link Extra for Lotus Notes
3. LotusScript Extension:

   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.

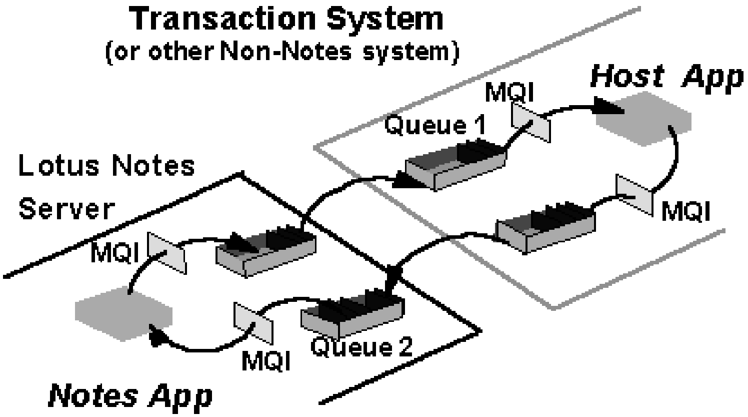
MQSeries allows Notes to participate in transaction systems by allowing Notes to initiate transactions and by acting as a store-and-forward data repository for data from transaction systems. The enterprise critical data and the associated business rules continue to be managed by the transaction system. 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
Transaction Systems and MQ
MQSeries link or CICS link
Notes Server
Mobile Notes User
Host-based Systems
MVS/ESA, (CICS/ IMS), VSE/ESA (CICS), OS/2, Windows/NT, OS/400, AIX/6000, HP-UX, SunOs & Solaris, SCO Unix, Unixware, DEC, Tandem

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.

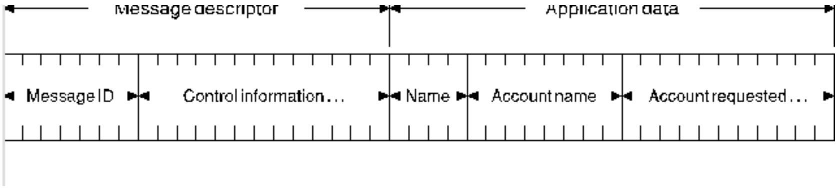
  MQSeries allows access to enterprise *processes,* whereas the other techniques allow access to enterprise data. Using MQSeries, the *only* way to access data is through a transaction, never directly. This may be considered limiting in that programs must be used to access data. Yet, it is enabling in 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.

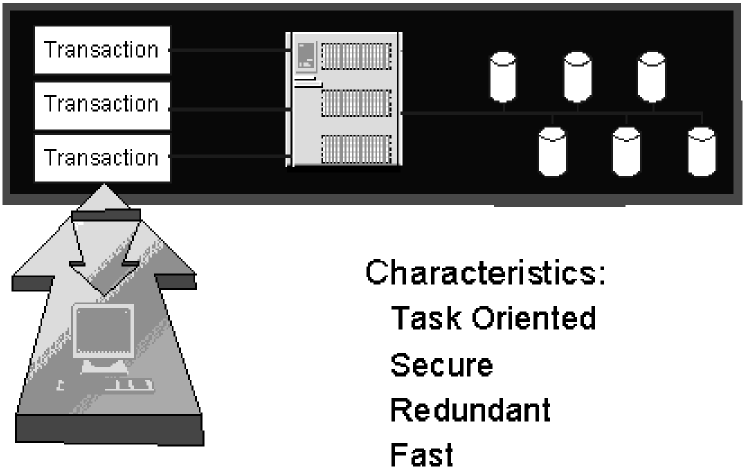
Queue managers

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.

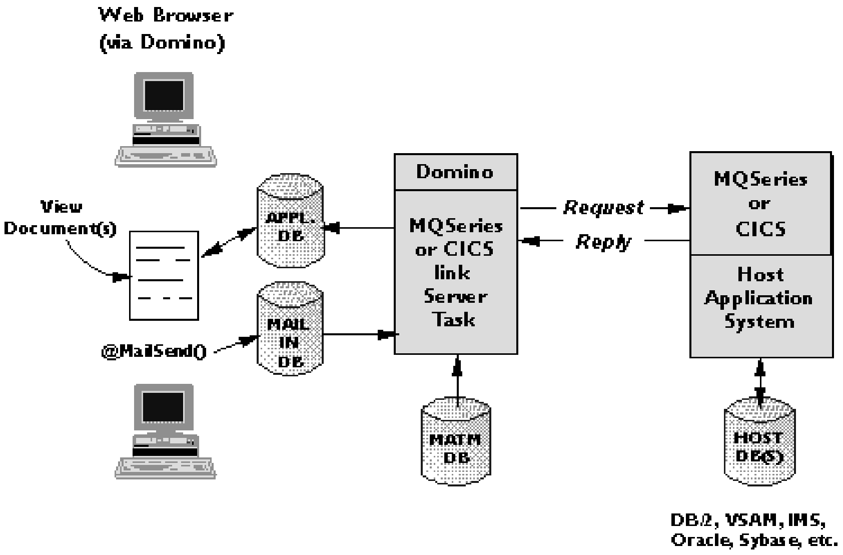
The actual link and switching process is managed by controlling the MATM database. This is a Notes database using standard Notes conventions. It controls the link that is a Notes server add-in task, so its management is most appropriately performed by the Notes Administrator.

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:

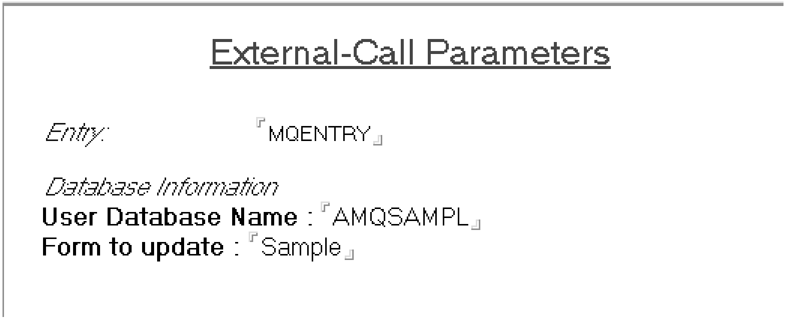
1. MQSeries

   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

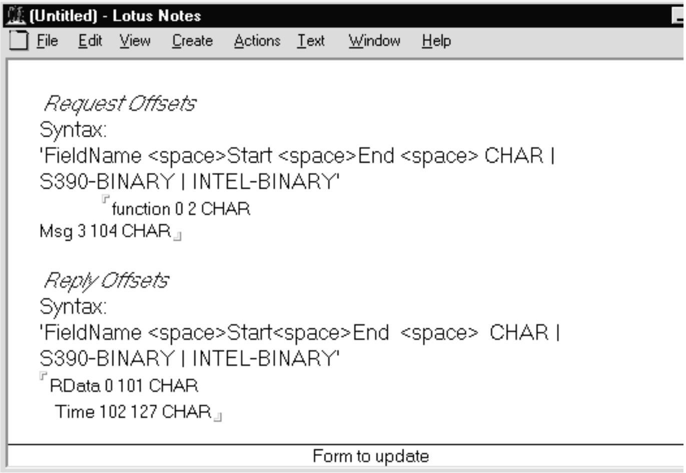
Entry: ⌜MQENTRY⌟

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.



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:

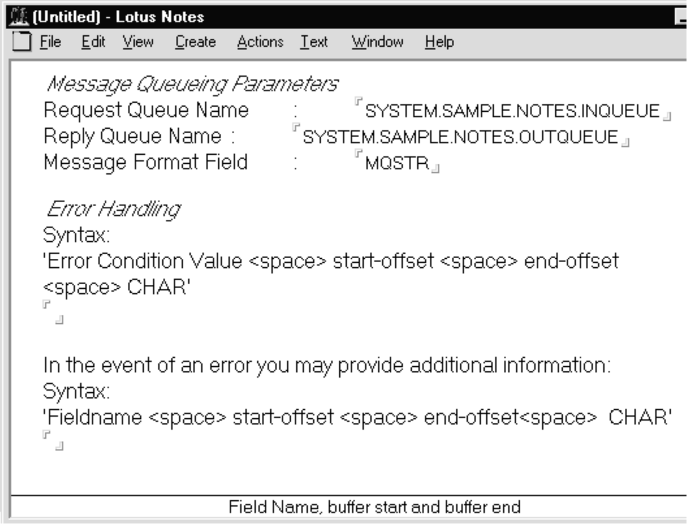
'FieldName <space>Start<space>End <space> CHAR | S390-BINARY | INTEL-BINARY'

```
⌜RData 0 101 CHAR
  Time 102 127 CHAR⌟
```

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

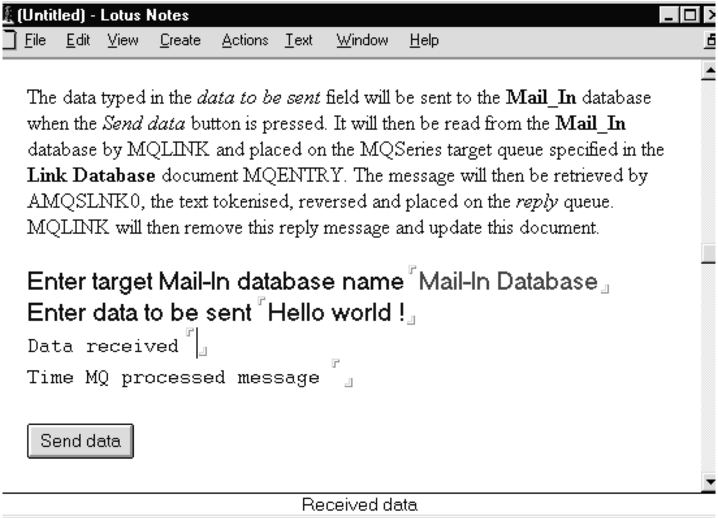
Message Queueing Parameters
Request Queue Name : SYSTEM.SAMPLE.NOTES.INQUEUE
Reply Queue Name : SYSTEM.SAMPLE.NOTES.OUTQUEUE
Message Format Field : MQSTR

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

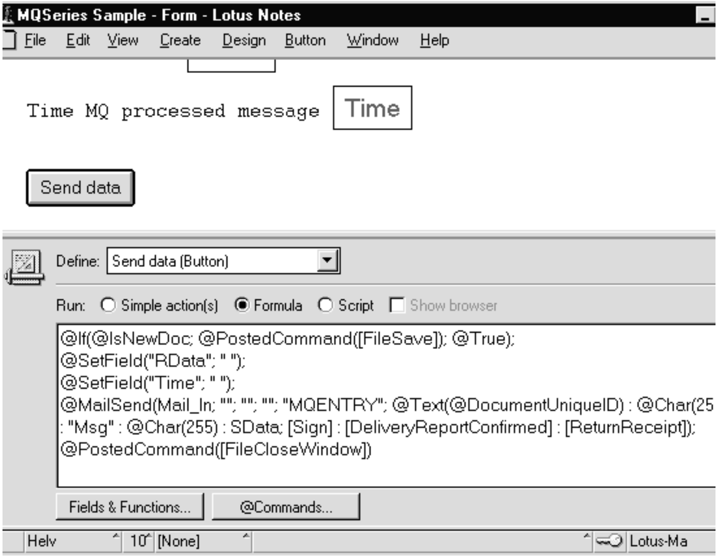
9. Set up your application to create mail entries to your Mail_In database as specified below. A button can be placed on a form of your application to send data to MQSeries as in the sample application:



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

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


MQSeries Sample - Form - Lotus Notes
File Edit View Create Design Button Window Help
Time MQ processed message
Time
Send data
Define: Send data (Button)
Run: ○ Simple action(s) ● Formula ○ Script □ Show browser
Fields & Functions...
@Commands...
Helv
10
[None]
Lotus-Ma

```
@If(@IsNewDoc; @PostedCommand([FileSave]); @True);
@SetField("RData"; " ");
@SetField("Time"; " ");
@MailSend(Mail_In; ""; ""; ""; "MQENTRY"; @Text(@DocumentUniqueID) : @Char(25
: "Msg" : @Char(255) : SData; [Sign] : [DeliveryReportConfirmed] : [ReturnReceipt]);
@PostedCommand([FileCloseWindow])
```

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:

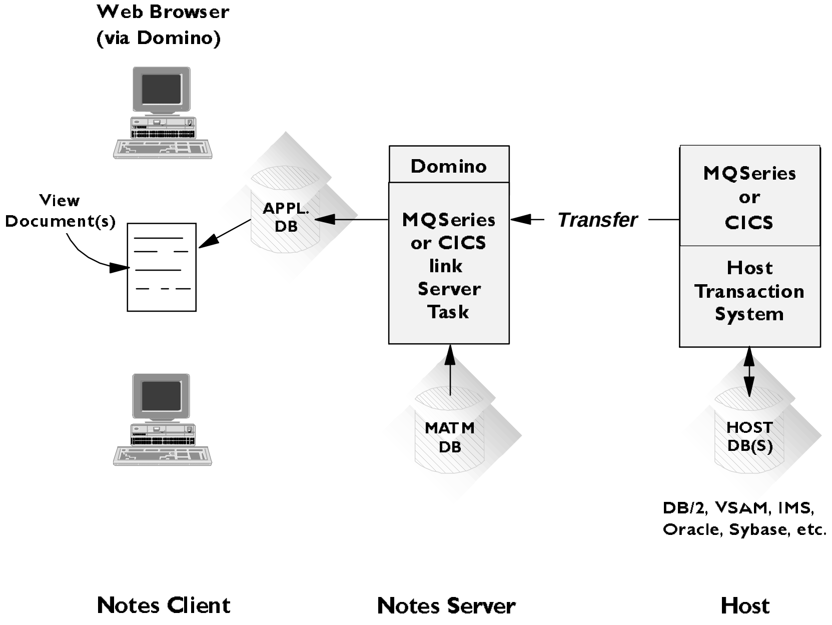
- The primary rule used to define how the data is processed given existing documents in the Lotus Notes database
- An optional Additional Selection Formula used to specify search criteria to be used against the Notes database

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.



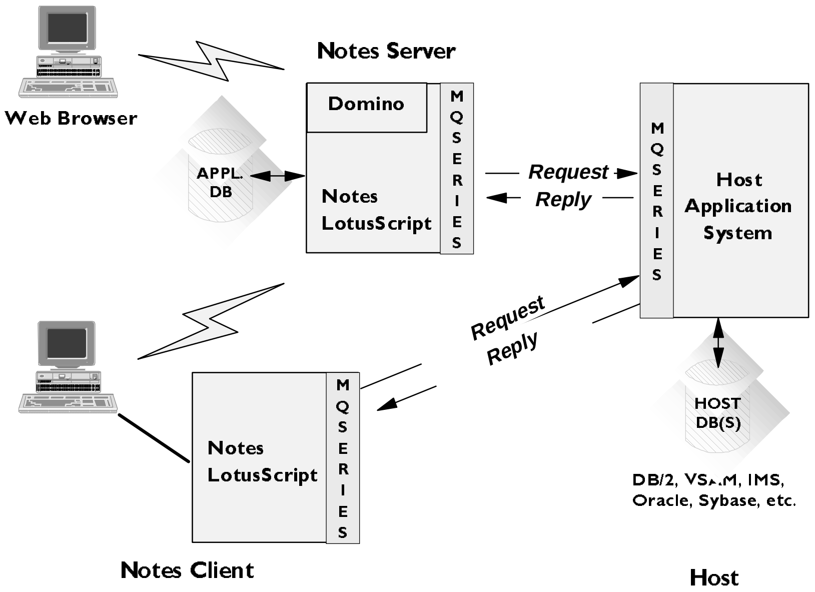
The MQSeries Link LotusScript Extension (MQLSX)

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.

- MQQueue

  This represents a connection to an MQSeries Queue. This connection is provided by an associated MQQueueManager object. When an object of this class is destroyed, it is automatically closed.

- MQQueueManager

  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 Mqqms As New MQSession
           Dim Mqqmgr 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
```

Chapter 20
Accessing Notes With the Notes C++ API

Lotus Notes contains an open interface to enable you to access Notes objects from external applications. In this chapter, we will describe the Notes C++ API, an easy-to-use class library. You will learn how to use it in C++ programs to access Notes facilities.

Any information in this chapter relates to the C++ API Beta 1 Release which was the current version at the time of this writing.

Overview

The Notes C++ API is a C++ library consisting of a set of classes that enable you to write application programs to create, access, and manage Notes Release 4 databases. It provides you with an object-oriented interface to Lotus Notes built on the Notes C API, and provides easier, more consistent access to Notes functionality than the C API.

You can use the Notes C++ API to create a variety of applications. You can write application programs that

- Retrieve data from Notes databases.

  Your application programs get access to yet another data source, just like they may have access to file systems or RDBMS, for example.

- Transfer information between Notes and non-Notes databases.

  For example, a program can exchange data between Notes and an object-oriented database or vice versa.

- Create Notes databases that are a combination of existing databases.

  You can collect and analyze data from multiple databases, and store the results in a higher-level strategic information database.

- Perform periodic maintenance of Notes databases.

  For example, a program can scan a Notes mail database and delete every message more than six months old.

Types of Applications

Programs written using the Notes C++ API may take a variety of formats. You can package C++ API code into stand-alone applications, dynamic libraries, and Notes server add-in tasks.

In all these program formats, the core API code is essentially the same. You use the same C++ API functions to operate on Notes databases. Only some additional API calls have to be added to control server add-ins.

Stand-alone Applications and Dynamic Libraries

Stand-alone API applications are main programs that make API calls into the Notes software. So, you must have Notes installed on the machine where these programs run, but you do not need to run Notes workstation software before running API programs.

Dynamic libraries are triggered by calls to their exported functions, and can call any API function.

Notes Server Add-In Tasks

The Notes server software is composed of standard tasks that carry out the server functions. The C++ API lets you extend the Notes software with server add-ins. A Notes server add-in is a custom server task that runs alongside the standard Notes server tasks.

Contents of the Notes C++ API Distribution

At the time of this writing, the Lotus Notes C++ API was available for OS/2 Warp, Windows 3.X, Windows 95, Windows NT, and AIX 4.1.

For each of those platforms, the Lotus Notes C++ API contains

- Documentation.

  Two Notes databases contain the *API User Guide* and *Reference Guide*. The latter contains detailed descriptions of all API classes, member functions as well as global functions, macros, enumerations, and constants.

- Sample programs.

  The distribution contains source code to a variety of sample programs. It also provides makefiles for the supported software platforms, and some sample Notes databases to demonstrate the capabilities of the programs.

- Development files.

  These files comprise internal header files that are included in Notes C++ API source modules, library files that you link to your Notes C++ API programs, and Notes C++ include files.

  API programs written for different platforms share the same header files, whereas library files are platform-specific.

Note The only include file you need for your programs is LNCPPAPI.H.

The Notes C++ API Architecture

This section presents you with an overview of the C++ API classes and data types.

Built-In Data Types

In order to enhance the portability of C++ API programs, the C++ API includes several macros for basic data types. You are encouraged to use them rather than the native C++ data types.

- LNBOOL

  Variables of this type take one of the constant values: FALSE (equals 0), or TRUE (different from 0).

- LNINT

  A 32-bit signed integer type.

- LNSINT

  A 32-bit unsigned integer type.

- LNNUMBER

  A 64-bit IEEE real number type.

- LNSTATUS

  A 32-bit error status type returned by C++ API functions. The constant value LNNOERROR (equals 0) stands for success; a non-zero value indicates an error or warning condition.

Note The API establishes certain naming conventions, because some compilers haven't yet implemented the C++ namespace constructs. The names of classes, global functions, and macros are prefixed with "LN", and they are in propercase, where each word begins with an uppercase character. Enumerations and data types also begin with the "LN" prefix, but contain only uppercase characters.

Common Classes

The C++ API library includes certain classes for dealing with data types such as numbers, date and time information, strings, and arrays of strings. You can use the classes as-is to handle those types; later on you will notice that they are exactly the classes that represent the item values stored in Notes documents.

For a description of the class member functions, refer to the *API Reference Guide*.

The LNNumber Class

The LNNumber class represents an LNNUMBER and provides member functions to convert strings into numbers and vice versa.

The LNNumbers Class

This class represents an array of LNNumber objects. It provides you with member functions to copy such arrays, and to insert, append, update and delete members.

The LNString Class

The LNString class facilitates the string handling in your API programs. It represents a null-terminated LMBCS string you can manipulate in many different ways. For example, it contains member functions for character, substring, and word handling as well as operators to compare strings and to extract characters.

The LNText Class

This class represents an array of LNString objects. It provides you with the same functionality for array handling as the class LNNumbers does.

The LNDatetime Class

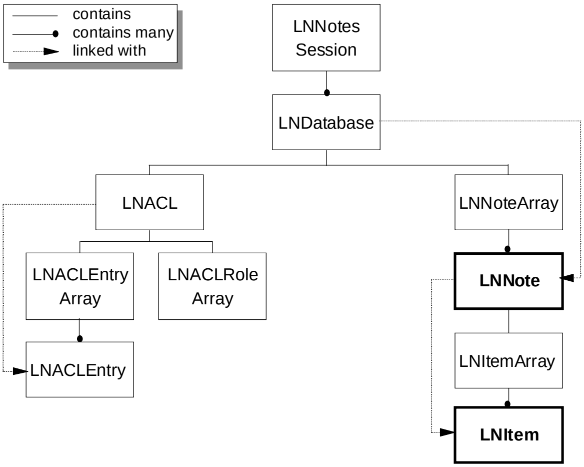
Date and time information is represented by this class. The LNDatetime class contains comparison operators and member functions to retrieve and set the individual date and time components.

C++ Class Hierarchy

The Notes C++ API consists of a set of classes, each of which gives you access to a specific type of Notes objects. The design of these classes reflects the Notes object properties and relationships between objects as they can be viewed from a conceptual point of view. For example, Notes databases have a server property, and contain agents and documents. Again, documents have a form property, and possibly responses, and in turn consist of a set of items.

This leads naturally to a hierarchy of containment relations between the API classes. Likewise, there is also an inheritance hierarchy. For example, consider items which store a list of values. The name and value type properties are common to all items, but the actual list of values is specific for each value type.

The Containment Hierarchy



The top-level class is LNNotesSession. At the beginning of any C++ API program, you create an LNNotesSession object in whose context you create, delete, and access one ore more Notes databases. LNDatabase objects contain an ACL and an array of LNNote objects.

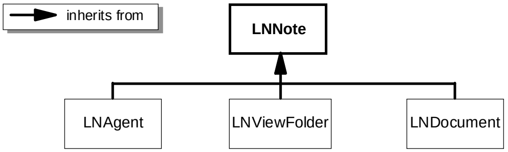
A database object contains exactly one object of class LNACL. This class in turn contains a set (implemented as array) of applicable roles, and a set of ACL entries. The latter defines the access level for persons, servers, and groups of persons or servers.

On the other hand, a database object contains a set of notes. A note is the general term for all objects contained in a database. Thus, a note can represent different types of objects, such as agents or documents. Common to all notes is that they in turn contain a set of items. More specific operations on a note require explicit knowledge of its type; so they are defined using inheritance. For example, a document can have responses whereas an agent can not.

So, this hierarchy defines how to navigate through the Notes object space. However, for some operations it is more convenient to use fast links to sub-objects. For example, consider how to access a particular item of a document. This is achieved by additional references as indicated in the figure above.

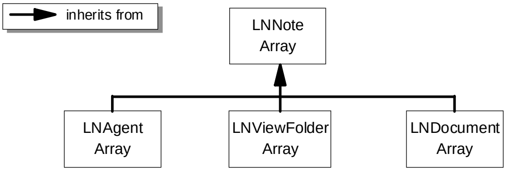
The Inheritance Hierarchies

The classes for the actual Notes database objects are derived from the class LNNote. This base class provides common member functions to work with the items contained in a note; the derived classes LNAgent, LNDocument, and LNViewFolder represent the Notes objects. Indeed, it is unlikely that you ever create an object of the class LNNote, because it is not related to any existing Notes object.

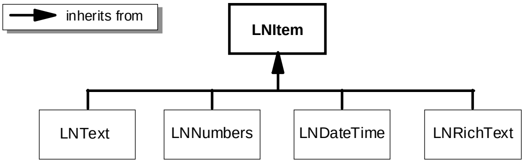


As depicted in the containment hierarchy, objects of the class LNNote are contained in LNNoteArray objects. But notes are always either agents, documents, views, or folders, and therefore an LNDatabase object will never return a set of LNNote objects but rather a set of agents, documents, and so on.

So, the classes LNAgentArray, LNDocumentArray, and LNViewFolderArray are derived from LNNoteArray. They use the functionality of the base class to provide type-safe access to the contained agents, documents, folders, and views. This means for example that member access functions of the class LNDocumentArray return an LNDocument object instead of an LNNote object.



The same considerations apply to items stored in a note. All items share some common properties and functionality, and there are objects capable of maintaining a set of items, namely of class LNItemArray. The following figure shows the item classes actually contained in documents.



Note The common classes are derived from the class LNItem. Furthermore, there is a class LNRichtext that is able to represent rich text items.

Error Handling

Most of the API class member functions return an LNSTATUS status code to indicate success or failure. Those that don't return such a code use the C++ exception mechanism to indicate a failure, by throwing an LNSTATUS exception. To handle C++ exceptions, your program uses the try and catch blocks, as illustrated below.

```
try
{
   // program statements
}
catch(LNSTATUS error)
{
   //error-handling statements
}
```

Note You may want to provide at least one pair of try/catch blocks in your main function, so that if an exception is thrown, your program will be able to clean up as needed before terminating.

Because most API functions return an LNSTATUS value, your program would typically test the return value of each function call to detect errors. This provides your program with a lot of control over error handling, but it also makes the program more cumbersome to write. In many applications, you might prefer to handle API errors centrally, in one or more places in your program, rather than testing the return value of each function.

The C++ API global LNSetThrowAllErrors function allows you to do this. The statement LNSetThrowAllErrors(TRUE) instructs the API to throw an LNSTATUS exception whenever an API error occurs anywhere in your program. This means your program can use C++ try/catch blocks to handle all API errors, rather than testing return codes for individual functions. This feature remains in effect until you disable it using LNSetThrowAllErrors(FALSE). You can use the LNGetThrowAllErrors function to test whether the feature is enabled or disabled; the default is disabled.

Note Some functions also return warnings in the LNSTATUS values. Those are never thrown as exceptions even if the exception mechanism is enabled.

Retrieving Error Messages

When an error or warning occurs, your program can initiate the appropriate error recovery, such as simply displaying an error message and returning.

LNSTATUS error codes have associated error messages, so you can retrieve the appropriate message for a given error. To do so, you call the global function

```
LNINT LNGetErrorMessage(LNSTATUS error, char *buf,
                        LNINT buflen =
LNERROR_MESSAGE_LENGTH)
```

You pass the error code in the first argument. The function retrieves the null-terminated error message in the buffer specified by the second argument. Optionally, you specify the buffer length in the third argument, which defaults to LNERROR_MESSAGE_LENGTH (512 bytes). If the message is longer than the buffer, LNGetErrorMessage truncates the message.

A Guided Tour Through the API

This section introduces the C++ API classes, and shows how to use them in application programs. It does not cover all classes or all member functions of the classes presented. So, if you miss a function you need for a particular task, refer to the *API Reference Guide*. It contains a complete list of them all.

Setting Up an Application Profile

Regardless of whether the C++ API program is a stand-alone program, a dynamic library, or a server add-in task, it must initiate a Notes session before using any API functions, and end the Notes session before terminating.

The C++ API is capable of supporting both single- and multi-threaded applications. For the latter, please refer to the *API User Guide* to see how they are set up.

Creating Single-Threaded Applications

To create a single-threaded application, you initiate a Notes session by creating an LNNotesSession object and calling the Init member function, and you end the session by calling the Term member function.

Note For UNIX applications only, the Init function requires the command line information that was passed to your program's main function.

The following code demonstrates the basic structure of a single-threaded C++ API program, but without error checking:

```
void main(int argc, char *argv[])
{
   LNNotesSession session;// create session object on stack
   session.Init();   // for UNIX: session.Init(argc, argv);
   // ... your code ...
   session.Term();
}
```

The NotesSession Class

The LNNotesSession class provides an application program with a connection to Notes. Its member functions can be divided into two groups: functions to access databases, and some general purpose functions.

Note If you are familiar with the LotusScript Notes class NotesSession, you will notice the similarities between them.

The following functions are not within the scope of a Notes database:

- Retrieving the local Notes data directory.

  LNString GetDataDirectory ()

- Retrieving the user name from the ID file on the local machine.

  LNString GetUserName ()

- Environment variable handling.

  The two functions

  LNString GetEnvironmentString (const LNString &variable)

  and

  ```
  LNSTATUS SetEnvironmentString (const LNString &variable,
                                 const LNString &string)
  ```

  allow you to return and set the values of environment variables as defined in the local NOTES.INI file.

- Current date and time.

  LNDatetime GetCurrentDatetime ()

The following functions allow you to create, delete and access databases:

- Open an existing database.

  ```
  LNSTATUS GetDatabase (const LNString &path,
                        LNDatabase *db,
                        const LNString &server = “” )
  ```

Given the path and the server of the database to be opened, this function initializes a database object whose address is the second parameter.

Note Whenever a function returns an object as result of the operation, it is a pointer type argument to the function. In that case, you have to pass the address of an existing object of that type that will be initialized on return.

- Create and open a new database copy.

  As for many other class member functions, this function exists in several versions allowing you to use the simplest one that fits your needs. The version with the most parameters is the following:

```
LNSTATUS CreateDatabaseCopy (const LNString &srcdb_path,
                             const LNString &srcdb_server,
                             const LNString &newdb_path,
                             const LNString &newdb_server,
                             constLNCreateDatabaseOptions
                                            &options,
                             LNDatabase *newdb = 0 )
```

  You specify the Notes server and the path of the source database, the server and the path of the database to be created, and a set of options for the new database. In the last argument you pass the address of an LNDatabase object that is initialized on successful return.

- Create and open a new database from a template.

  Again, the function CreateDatabaseFromTemplate is overloaded. There is a version that has exactly the same arguments as the function CreateDatabaseCopy shown above. Here, the first two arguments describe the server and path of the database template.

- Delete a database.

```
LNSTATUS DeleteDatabase (const LNString &path,
                         const LNString &server  = “”)
```

  This version of the DeleteDatabase function gets the server and the path of the database in question.

- Access an existing database

```
LNSTATUS GetDatabase (const LNString &path,
                      LNDatabase *db,
                      const LNString &server = “” )
```

  This function initializes the database object whose address is passed as second argument, with the database at the specified location.

Note As you may have noticed, a Notes database is not created by instantiating a new object of class LNDatabase, and is not deleted by a corresponding delete operator, but rather by function calls to a session object. This is due to the general API concept to use these classes as easy-to-use interfaces to the real Notes objects. As such, an LNDatabase object is not a database itself; think of it as a handle to a database. The same applies to all other objects stored in databases.

Example: Creating a New Database

Now, we will use the member functions of the LNNotesSession class to create a discussion database that will be used throughout the rest of the chapter. The example also shows how to initiate a session, and to set up an error handler.

```
#include <iostream.h>
#include <lncppapi.h>
void main(int argc, char *argv[])
{
  LNNotesSession session;           // Create a session
object   LNDatabase discussDB;             // Create a
database object   LNCreateDatabaseOptions options;  // Create
an options object
  LNSetThrowAllErrors(TRUE);        // Enable the API
exception
                                    // mechanism
  try
  {
    session.Init();

    // Set options: the new database will inherit design
    // updates
    options.SetInheritDesign (TRUE);

    // Create a database from the discussion template on the
    // local machine. It will be located at the Notes data
    // directory.
    session.CreateDatabaseFromTemplate ("DISCUSS4.NTF",
                                          "",
                                            "DISCUSAPI.NSF",
                                              "",
                                                options,

&discussDB);

    discussDB.Close();   // close the database
  } // end try
  catch(LNSTATUS error)
  {
    char errorBuf[LNERROR_MESSAGE_LENGTH];

    LNGetErrorMessage(error, errorBuf);
    cerr << "Error:  " << errorBuf << endl;
  }
}
```

Working With Databases

The LNDatabase class provides access to all objects contained in a database: agents, views, folders, and documents. Furthermore, you can access and modify the database properties, such as the name of the database, and the ACL.

Opening and Closing a Database

Before you can perform any operations on a database, you must open it. Creating a database with LNNotesSession::CreateDatabase automatically opens the database, whereas a call to LNNotesSession::GetDatabase does not.

- Opening a database.

  `LNSTATUS Open (LNDBOPENFLAGS flags)`

  The passed optional argument specifies how operations on the database should be performed. The only flag you can specify is LNDBOPENFLAGS_DELAY_COMMIT. Setting this flag means that all updates will take place in cache memory; when you close the database, its contents on disk are synchronized.

  Tip Using this flag will speed up your API operations, but if the machine crashes, the changes are lost.

- Closing a database .

  `LNSTATUS Close ()`

  Closing a database also closes any notes it contains.

  Note Any unsaved changes you made to any notes in the database will be lost when you close the database, so be sure to save any changes first. Also, if you don’t close a database object before it goes out of scope, it will close automatically.

Accessing Database Properties

- Retrieving the Notes server and the path of a database.

  ```
  LNString GetServer ()
  LNString GetFilepath ()
  ```

  If the database is stored on the local machine, GetServer returns an empty string.

- Accessing the database title.

  LNString GetTitle ()
  LNSTATUS SetTitle (const LNString &title)

- Accessing the design template attributes.

  LNString GetTemplateName ()

  It returns an empty string if the database is not a template.

  LNSTATUS SetTemplateName (const LNString &name)

  This function defines a new name for this database template.

  LNString GetInheritsFromTemplateName ()
  LNSTATUS SetInheritsFromTemplateName (const LNString &name)

  You can use these functions to determine whether the database is based on a template, and you can assign a new database template to it.

- Accessing the database ACL.

  LNSTATUS GetACL (LNACL *acl)

  This function initializes the ACL object whose address you pass as argument.

Continuing with our sample database, we will now assign a meaningful title to it, and define a different default access right.

```
LNNotesSession session;      // Create a session object
LNDatabase discussDB;        // Create a database object
LNACL acl;

LNSetThrowAllErrors(TRUE);        // Enable exceptions
try
{
   session.Init();

   // Get the database and open it
   session.GetDatabase ("DISCUSAPI.NSF", &discussDB, "");
   discussDB.Open ();

   // Set the title
   discussDB.SetTitle ("Notes C++ API Discussion");

   // Get the ACL
   discussDB.GetACL (&acl);

   // Set the default access to "no access"
   acl.SetDefaultAccessLevel (LNACLLEVEL_NO_ACCESS);
```

```
      // Save the change
      acl.Save ();

      // Close the database
      discussDB.Close()

   }
```

Accessing Views and Folders

In order to access database views and folders, an LNDatabase object provides the following functions:

- Accessing a view or folder by name.

  ```
  LNSTATUS GetViewFolder (const LNString &viewname,
                          LNViewFolder *view)
  ```

  This function initializes the view object whose address is passed as argument, to the view or folder with the specified name.

- Accessing all views and folders.

  ```
  LNSTATUS GetViewFolders (LNViewFolderArray *views)
  ```

- Test whether a particular view or folder exists.

  ```
  LNBOOL ViewFolderExists (const LNString &viewname)
  ```

Accessing Documents

The LNDatebase class provides functions to create, delete, and access documents. Most of them either require an LNDocument object as parameter, or they return such an object. Again, whenever a function returns a database object, you pass the address of the object to be initialized.

- Create new documents.

  ```
  LNSTATUS CreateDocument (LNDocument *newdoc)
  LNSTATUS CreateDocument (LNDocument *newdoc,
                           const LNString &formname)
  ```

  The first function creates a new blank document with the default form. The new document is returned in newDoc.

- Create copies of existing documents.

  ```
  LNSTATUS CreateDocument (LNDocument &document,
                           LNDocument *newdoc)
  ```

  This function takes document and creates a new copy of it that is returned in newDoc.

- Delete documents.

  ```
  LNSTATUS DeleteDocument (LNDocument *document)
  ```

- Get all documents in the database.

  ```
  LNSTATUS GetDocuments (LNDocumentArray *docs)
  ```

  This function initializes the document array whose address is passed as argument, with all documents in the database.

Searching for Documents

When you have to search for the documents you want to access, you have basically two choices. Either you use one of the following search functions as provided by a database object, or you navigate through a view or folder.

- Selective search for documents.

  ```
  LNSTATUS Search (const LNString &formula,
                   LNNoteArray *results)
  LNSTATUS Search (const LNString &formula,
                      LNNoteArray *results,
                      LNSearchOptions *options)
  ```

  These are general search functions that allow you to search for any notes in the database, including agents, views, folders, and documents. The selection criteria are expressed by a formula string that can include any of the Notes @ functions, field names, and logical operators.

  With the search options, you can restrict the search to notes modified during a given period of time. Moreover, you can specify only to retrieve a certain note type such as documents, for example.

- Full-Text search for documents is also available. Refer to the *API User Guide* for details of setting up a full-text index and performing a full-text search.

In our sample discussion database, we will now delete all documents that were created by a particular person, for example. Here’s the code:

```
  LNNotesSession session;           // Create a session
object
  LNDatabase discussDB;             // Create a database
object
  LNSearchOptions options;          // Create a search
options
                                    // object
  LNDocumentArray resultDocs;       // Create a search result
                                    // container
  LNINT i;

  LNSetThrowAllErrors(TRUE);        // Enable exceptions
  try
  {
     session.Init();
```

```
     // Get the database and open it
     session.GetDatabase ("DISCUSAPI.NSF", &discussDB, "");
     discussDB.Open ();

     // Specify in the options that we are only interested
       // in notes of type documents within the specified
period
     // of time.
     options.SetNoteType (LNNOTETYPE_DOCUMENT);
     options.SetBeginDate ("01/01/1989");
     options.SetEndDate (session.GetCurrentDatetime());

     // The query is: Find all documents whose author
     // is Henry Miller.
     discussDB.Search (
            "@Name( [CN]; @Author) = \"Henry Miller\"",
                       &resultDocs,
                       &options);

     // Delete all found documents (arrays start at index 0)
     for (i = 0; i < resultDocs.GetCount(); i++)
        discussDB.DeleteDocument (&resultDocs[i]);

     // Close the database
     discussDB.Close();
  }
```

Working With Documents

Once you have got an LNDocument object, you can can retrieve and modify all its properties and items.

Opening, Saving, and Closing Documents

You get a document of a Notes database either by creating it, or as a result of a database search. Creating a document implicitly opens it. In all other cases, you need to open it before you can access its properties and items. Likewise, after you have performed all changes, you need to call the Save method before closing the document. Otherwise all changes are lost.

The functions are the following:

- Opening a document.

  ```
  LNSTATUS Open (LNNOTEOPENFLAGS flags=
                                   LNNOTEOPENFLAGS_DEFAULT )
  ```

  In the flags, you can specify not to mark the document as read, for example.

- Saving a document.

  ```
  LNSTATUS Save (LNNOTESAVEFLAGS flags=
                                  LNNOTESAVEFLAGS_DEFAULT )
  ```

  In the flags, you can specify options such as delayed writing, or forced saving.

- Closing a document.

  ```
  LNSTATUS Close ()
  ```

Creating, Deleting, and Accessing Items and Their Values

These are the functions to retrieve items of a documents, and to create new ones.

- Creating new items.

  ```
  LNSTATUS CreateItem (const LNString &name,
                       LNItem *newitem,
                       LNITEMFLAGS flags = 0,
                       LNITEMOPTIONS options =
                               LNITEMOPTIONS_DELETE_APPEND)
  ```

  This function creates the new item with the specified name in the document. In the flags, you can specify whether the item shall be signed, encrypted, or protected. With the options argument, you can influence the operation behavior by specifying what action should take place when an item with this name already exists: append the item, delete the existing item first, or treat this case as an error.

  ```
  LNSTATUS CreateItem (const LNItem &item,
                       LNITEMOPTIONS options =
                               LNITEMOPTIONS_DELETE_APPEND,
                       LNItem *newitem = 0)

  LNSTATUS CreateItem (const LNString &name,
                       const LNItem &item,
                       LNITEMFLAGS flags = 0,
                       LNITEMOPTIONS options =
                               LNITEMOPTIONS_DELETE_APPEND,
                       LNItem *newitem = 0 )
  ```

  These functions copy an existing item into the document. The second version allows you to specify a name different from the source item.

- Deleting items.

  ```
  LNSTATUS DeleteItem( const LNString &name )
  LNSTATUS DeleteItem( LNItem &item )
  ```

- Retrieving existing items.

  LNSTATUS GetItem (const LNString &name, LNItem *item)

  This function gets the specified item within the note. The input is a string that contains the name of the item.

  LNINT GetItemCount ()

  Returns the number of items within the note.

  ```
  LNSTATUS GetItems (LNItemArray *items,
                     LNITEMTYPE type = LNITEMTYPE_ANY)
  ```

  This function gets all items of a document.

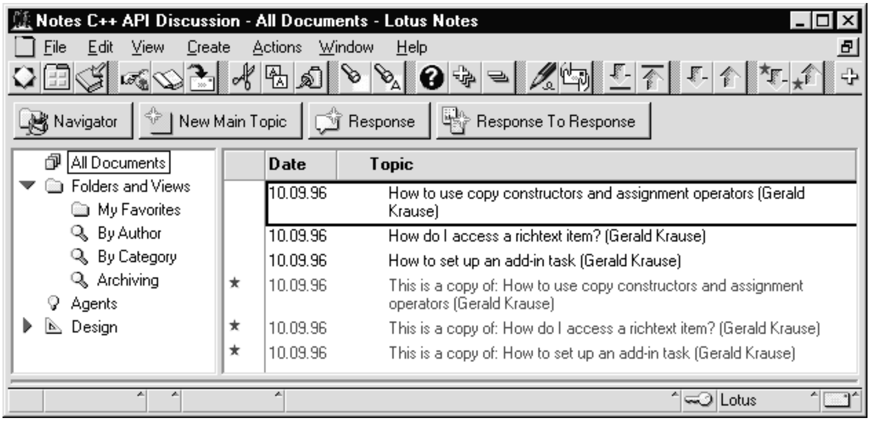
Note Although all these declarations expect LNItem objects as input and output argument types, you will never pass such an object but rather an object of a class derived from it, such as a LNText, LNNumbers, or LNRichtext.

Now, we are ready to automatically create some documents in our discussion database. You should notice how the common class types such as LNText are used to work with the actual item values. In this example, we simply copy all documents of the database, and modify their subjects.

```
LNNotesSession session;      // Create a session object
LNDatabase discussDB;        // Create a database object
LNDocumentArray documents;   // Create a document container
LNINT i;

LNSetThrowAllErrors(TRUE);   // Enable exceptions
try
{
   session.Init();

   // Get the database and open it
   session.GetDatabase ("DISCUSAPI.NSF", &discussDB, "");
   discussDB.Open ();

   // Get all documents of the database
   discussDB.GetDocuments (&documents);

   // For all these documents:
   // first, create copies of them in the database,
   // and then modify their subject item
   for (i = 0; i < documents.GetCount(); i++)
   {
      LNDocument srcDoc = documents[i];
      LNDocument newDoc;
      LNText subjectItem;
      LNString newSubject;
```

```
      // Create a new copy of the i-th document
      // in the database
      // *** DO NOT USE THE COPY CONSTRUCTOR! ***
      discussDB.CreateDocument (srcDoc, &newDoc);

      // Get the title item of the new document
      // and change it (the index 0 of the string array!)
      newDoc.GetItem ("Subject", &subjectItem);

      newSubject = "This is a copy of: ";
      newSubject += subjectItem [0];
      subjectItem[0] = newSubject;

      // Save the change
      newDoc.Save ();
   }

   // Close the database
   discussDB.Close();
}
```

To give you an impression of how the result looks, here's a screen shot of the main database view after running the program:



	Date	Topic
	10.09.96	How to use copy constructors and assignment operators (Gerald Krause)
	10.09.96	How do I access a richtext item? (Gerald Krause)
	10.09.96	How to set up an add-in task (Gerald Krause)
★	10.09.96	This is a copy of: How to use copy constructors and assignment operators (Gerald Krause)
★	10.09.96	This is a copy of: How do I access a richtext item? (Gerald Krause)
★	10.09.96	This is a copy of: How to set up an add-in task (Gerald Krause)

Working With Response Documents

Many databases make use of response documents to create relationships between documents. For example, in the sample database, you can create responses to any of the discussed topics. The C++ API provides the following functions to work with response documents:

- Creating a response.

  LNSTATUS MakeResponse (const LNDocument &parent)

  This function makes the current document a response of the one specified in the argument.

- Retrieving responses.

  ```
  LNINT GetResponseCount ()
  LNSTATUS GetResponses (LNDocumentArray *responses)
  ```

  These functions return the number of response documents, and the response documents itself, respectively. They include only the immediate responses of the documents, not the responses to responses.

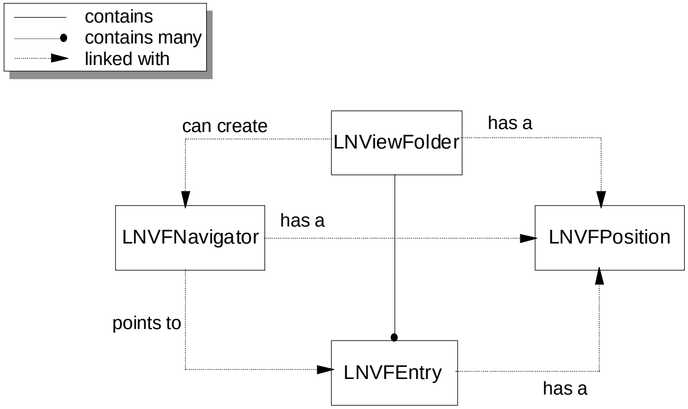
- Retrieving the parent documents.

  ```
  LNSTATUS GetParentDocument(LNDocument *parent)
  ```

  This function gets the parent of the document if it is a response document.

Working With Views and Folders

So far, you know that views and folders are represented by the class LNViewFolder which is a special kind of a note. For the following presentation, we need to refine the containment:



Folders and views are both collections of Notes documents. They can be treated in the same way, because they share the same design properties. So, your starting point is the class LNViewFolder which represents a view of folder. The entries (rows) of such objects are represented by LNVFEntry objects.

Views can contain main documents and response documents, and they can be organized by categories and sub-categories. Using the C++ API, you can access all or some entries in a collection. Objects of the class LNViewFolder provide you with many functions to navigate within the collection. However, if you want to access the responses (or more general, the children) or even several levels of descendants of an entry in the collection, you may want to instruct the LNViewFolder object to create a navigator object of the class LNVFNavigator. Although you could also use the

LNViewFolder object for that purpose, such a navigator object has the advantage that you don't lose the current position of the LNViewFolder object, and allows you to access entries at arbitrary levels. Furthermore, you can create multiple navigators to navigate through different collection areas simultaneously.

Regardless of whether you navigate through a collection with an LNViewFolder or LNVFNavigator object, the current position is represented by an LNVFEntry object. Of course, there is an exception: if the collection is empty, this entry doesn't exist. Each entry can provide you with its position in the collection which is represented by an LNVFPosition object. You can use such objects to compare the positions of different entries in the collection. The order is defined by the order in which they appear in the view or folder.

Opening and Closing View and Folders

You create an LNViewFolder object by calling the function LNDatabase::GetViewFolder. In order to access the collection, you then need to open it. When you don't need the collection any more, call the Close function for that object to release the associated resources.

- Opening a Collection

  ```
  LNSTATUS Open (LNVFOPENFLAGS flags =
  LNVFOPENFLAGS_DEFAULT)

  LNSTATUS Open (LNVFOPENFLAGS flags,
                 const LNDatabase &db )

  LNSTATUS Open (LNVFOPENFLAGS flags,
                 const LNString &pathname,
                 const LNString &server = “” )
  ```

  When you specify a database as argument, the view or folder is assumed to reside in that database, rather than the database containing this view/folder note.

- Closing a Collection

  ```
  LNSTATUS Close()
  ```

Navigational Functions

You can use an opened LNViewFolder object to navigate a view or folder in the following ways:

- Go to the first or last entry in the collection.

  ```
  LNSTATUS GotoFirst (LNVFEntry *entry = 0)
  LNSTATUS GotoLast (LNVFEntry *entry = 0)
  ```

- Go to the next or previous entry in the collection.

  ```
  LNSTATUS GotoNext (LNVFEntry *entry = 0)
  LNSTATUS GotoPrevious (LNVFEntry *entry = 0)
  ```

- Go to a main, parent, or child document.

  ```
  LNSTATUS GotoMain (LNVFEntry *entry = 0)
  LNSTATUS GotoParent (LNVFEntry *entry = 0)
  LNSTATUS GotoChild (LNVFEntry *entry = 0)
  ```

- Go to the first or last sibling having a common parent.

  ```
  LNSTATUS GotoFirstSibling (LNVFEntry *entry = 0)
  LNSTATUS GotoLastSibling (LNVFEntry *entry = 0)
  ```

  Note A sibling entry is one that has the same parent as the current entry.

There are even more functions to go to the next or previous category, non-category, main-topic, parent document, or sibling document. Furthermore, there is an extensive set of selective and full-text search functions for collection objects available. Please refer to the *API User Guide* for details.

All these functions affect the current position of the LNViewFolder object. If you pass the address of a collection entry object as argument, it is initialized with the current entry.

Caution Not all navigation functions are valid in all positions of the LNViewFolder object. Whenever a next or previous entry cannot be found, the warning status LNWARN_NOT_FOUND is returned. So, you should test the return value of these functions.

The current position of an LNViewFolder object is represented by an LNVFPosition object. Using the following functions, you can retrieve it, and you can use it to set the collection object to a particular position:

- LNSTATUS GetPosition (LNVFPosition *position)
- LNSTATUS SetPosition (const LNVFPosition &position)

As mentioned previously, you can also create a new independent navigator object. An LNViewFolder object provides the functions to initialize a navigator object with all collection entries, the immediate children, or all descendants of the current entry:

- LNSTATUS GetEntries (LNVFNavigator *navigator)
- LNSTATUS GetChildren (LNVFNavigator *navigator)
- LNSTATUS GetDescendants (LNVFNavigator *navigator)

Caution If the collection is a view, its contents are recomputed during object construction. This means that the original LNViewFolder object might contain different entries than the new navigator.

Accessing Collection Entries

Once you have positioned the LNViewFolder or LNVFNavigator object at an appropriate entry, you can retrieve its type, specific column values of that entry, and the complete document associated with it:

- Accessing a column either by position number or by name.

  ```
  LNItem LNVFEntry::operator [] (LNINT n)
  LNItem LNVFEntry::operator [] (LNString name)
  ```

  Some restrictions apply: you cannot access hidden columns; they are not counted for determining the position number. Some more restrictions apply to categorized columns; refer to the *API Reference Guide* for details.

- Accessing the entire document.

  ```
  LNSTATUS GetDocument (LNDocument *document)
  ```

- Retrieving the entry type.

  ```
  LNBOOL IsCategory ()
  LNBOOL IsMainTopic ()
  LNBOOL IsResponse ()
  LNBOOL IsTotal ()
  ```

Example

By accessing views and folders, we’re now able to perform some computations on our discussion database. In the following example, we compute the average number of responses per topic in the main view.

```
LNNotesSession   session;
LNDatabase       db;
LNViewFolder     view;
LNVFEntry        mainEntry;
LNINT            mainTopics = 0,
                 totalResponses = 0;

LNSetThrowAllErrors(TRUE);   // Throw all LNSTATUS errors
try
{
   session.Init();

   // Open the database
   session.GetDatabase("discusapi.nsf", &db);
   db.Open();

   // Get the "All Documents | ($All)" view and open it
   db.GetViewFolder("($All)", &view);
   view.Open();
```

```
   // Get the first main entry
   view.GetEntry(&mainEntry);

   // Get the number of entries at the same level
   mainTopics = mainEntry.GetSiblingCount();

   // Iterate over all main entries and count their
   // descendants.
   do
      totalResponses += mainEntry.GetDescendantCount();
   while(view.GotoNextMain(&mainEntry) != LNWARN_NOT_FOUND);

   // Print the average number of responses per topic
   if (mainTopics > 0)
      cout << "Number of Main Topics: " << mainTopics
           << endl
           << "Average Responses per Topic: "
           << (double) totalResponses / mainTopics << endl;
   else
      cout << "There is no discussion." << endl;
}
```

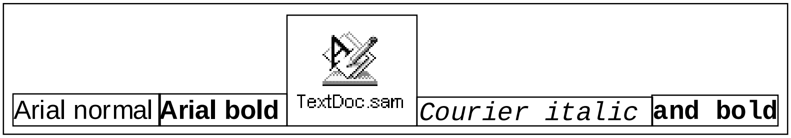
A Closer Look at Rich Text Items

Richtext items are the most powerful items that a Notes document can contain. They form a single big container that can store stylized text, tables, document links, OLE objects, file attachments, bitmaps, and a combination of all of these.

The richness of these items requires a finer grained structure for its representation than the class LNRichText provides. You will need to know about this representation, because it defines how you access individual elements in a richtext item.

The C++ Representation of Richtext Items

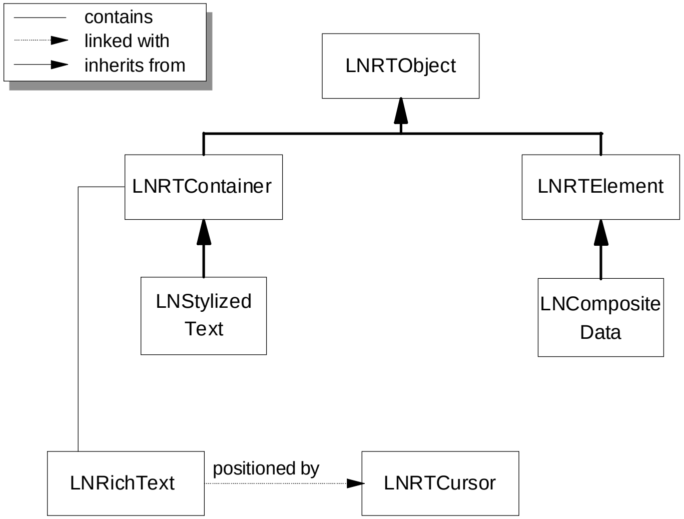
A richtext item can be considered as a sequence of containers of different types. The container type determines the elements in it. Before talking in terms of C++ classes, let us consider an example:



In this figure, the complete richtext item is a generic container that contains five other containers; four of them contain stylized text, that is text associated with exactly one style defined by a font, a size, and a face. In the middle is a file attachment, which is a so-called composite data element; it is enclosed in a generic container.

For all these container types, the C++ API defines separate classes representing the properties of the contained elements. When you access a richtext item, you will navigate through a sequence of such containers which is determined by the contents of the richtext item. The LNRichText class maintains the current position in the container by an object of the class LNRTCursor. You use this cursor object like a cursor of a text editor: it supplies functions to move to the next, previous, first, and to the last object in the richtext item.

The following figure shows the complete class hierarchy for richtext items:



The class LNRTObject is the base class for all other richtext container classes. It provides functionality common to all types of containers. The container objects that can currently appear in a richtext item, are objects of the classes LNStylizedText and LNCompositeData. It is planned to derive more classes from LNRTElement to enable the representation of tables and hotspots, for example.

The distinction between LNRTContainer and LNRTElement class objects influences the way in which you access them. An LNStylizedText object contains a sequence of characters that you can access individually by LNRTCursor movements. An object of a class derived from LNRTElement is indivisible.

Now, we will describe how you navigate through richtext using these classes. We have already described how you get access to a richtext item; in the same way as to any other item.

Navigating Through a Richtext Item

Given a richtext item, you call one of the following functions to obtain a cursor for that item:

- LNSTATUS GetCursor (LNRTCursor *cursor)
- LNSTATUS GetEndCursor (LNRTCursor *cursor)

  The first function returns a cursor that points to the first position in the item; the second form returns a cursor that points to the end of the last position.

Using this cursor object, you can now navigate through the container sequence:

- LNSTATUS GotoFirst (LNRTTYPE type, LNRTObject *object)
- LNSTATUS GotoLast (LNRTTYPE type, LNRTObject *object)
- LNSTATUS GotoNext (LNRTTYPE type, LNRTObject *object)
- LNSTATUS GotoPrevious (LNRTTYPE type, LNRTObject *object)

  The first argument for all these functions is the type of container you want to go to. You can specify one of the following values:

  - LNRTTYPE_STYLIZED_TEXT

    A stylized text container.

  - LNRTTYPE_COMPOSITE_DATA

    A composite data object such as a file attachment.

  - LNRTTYPE_CONTAINER

    Any container object; currently either a generic container or a stylized text container.

  - LNRTTYPE_ELEMENT

    Any indivisible richtext element; currently only a composite data object.

  - LNRTTYPE_OBJECT

    Any richtext object.

  You may notice, each of these values specify exactly one class position in the class hierarchy, together with all descendants of that class.

  The second argument to these functions is an optional output argument by which you access the richtext object at the new position.

There are many more navigation functions. For example, you can position the cursor by searching for a text string; refer to the *API User Guide* for details.

Accessing and Modifying Richtext Items

In the simplest form, you simply append some text or another richtext item to the richtext item using the LNRichText::Append function.

When you want to insert or delete some text or a composite object at a particular position in the item, do the following:

1. Use the navigation functions to position the cursor of the LNRichText object.
2. Retrieve the cursor object.
3. Call either the LNRichText::Insert, or the LNRichText::Delete function.

To give you a simple example, here's a program that opens the sample discussion database, retrieves the first document, and then inserts a stylized text at the third position:

```
LNNotesSession session;          // a session object
LNDatabase discussDB;            // a database object
LNDocumentArray documents;       // a document container
LNDocument doc;                  // a document
LNRichText rtItem;               // a richtext item
LNRTCursor rtCursor;             // a richtext cursor

LNSetThrowAllErrors(TRUE);       // Enable exceptions

try
{
   session.Init();

   // Get the database and open it
   session.GetDatabase ("DISCUSAPI.NSF", &discussDB, "");
   discussDB.Open ();

   // Get all documents of the database
   discussDB.GetDocuments (&documents);

   // Retrieve the first document and open it
   doc = documents[0];
   doc.Open ();

   // Access the Body richtext item
   doc.GetItem ("Body", &rtItem);

   // Get a cursor for it
   rtItem.GetCursor (&rtCursor);
   // Skip the first two richtext objects:
    // Ignore warnings that are returned when there is no
next
    // object.
    // If they do not exist, leave the cursor behind the last
    // object.
    rtCursor += 2;
    // equivalent to:
    //    rtCursor.GotoNext (LNRTTYPE_OBJECT);
    //    rtCursor.GotoNext (LNRTTYPE_OBJECT);

    // Now insert a new text
    rtItem.Insert ("The first insert!", &rtCursor);

    // save the changes
    doc.Save ();

    // Close the database
    discussDB.Close();
  }
```

Creating Notes Server Add-In Tasks

This section describes how to use the Notes C++ API to create custom Notes server add-in tasks.

What Are Server Add-In Tasks?

A server add-in task is an executable program that runs alongside other tasks that make up the Notes server software. It allows you to perform any operation on Notes databases accessible to the Notes server.

You may use server add-in tasks for a single operation, but its main purpose is to perform some operations periodically.

Note If you want to create a server add-in to perform a single operation, you may want to consider a stand-alone program as a well-suited alternative. Apart from the fact that a server add-in automatically logs in with the server account, there is no other difference.

For example, suppose that you want to retrieve the contents of a relational database periodically to update corresponding documents in a Notes database. You can do this with an add-in task as follows:

1. Develop a stand-alone program that contains all of the C++ API code needed to read the relational database and update the Notes database.
2. Add scheduling code to the program to extend it to a Notes server add-in. For example, you can specify the task to execute once per hour.
3. Start the add-in task with the Notes server software.

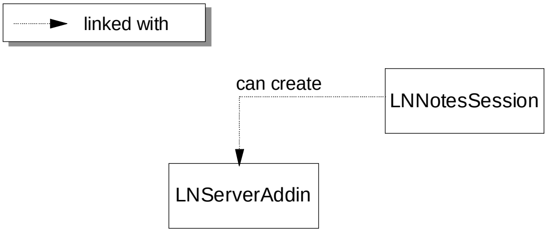
The add-in will run until the Notes server is shut down. Every hour, the add-in task will check the relational database for changes, and write these changes to the Notes database.

If some other time interval between executions of the task is required, the timing control code may be modified to specify any schedule, such as once each night at 2:00 AM, or once every 5 seconds. Add-ins may also be written to execute more than one job, each with its own schedule.

In the following, we will present you with the details of how to add the required scheduling code to an already existing C++ API program.

Program Structure of an Add-In Task

While the structure of an add-in task that executes only once is identical to a stand-alone API program, for periodic scheduling it is distinguished from other API applications by the presence of an LNServerAddin object, representing the add-in task. The following figure shows how this new class is related to those you already know:



In addition to the other API code you write to work with Notes, you will need to insert calls to control the execution of the server add-in.

First, declare an LNServerAddin object, and include one call to LNNotesSession::GetServerAddin, which actually retrieves the LNServerAddin object for your Notes session. The function GetServerAddin is declared as follows:

```
LNSTATUS GetServerAddin (const LNString &task,
                         const LNString &text,
                         LNServerAddin *addin)
```

The first and second arguments are the task name and description used for the default status line. If you issue a SHOW TASKS command at the server console, these arguments will be displayed together on a single line. The third argument is the address of an LNServerAddin object which will be initialized on return.

Once you have called GetServerAddin, you can call LNServerAddin functions as needed to control the execution of your server add-in task.

Controlling the Task Execution

The functions provided by the class LNServerAddin allow you to set up an appropriate schedule interval for your server add-in:

- Task scheduling.

  ```
  LNBOOL IsNewDay ()
  LNBOOL HaveMinutesElapsed (LNINT n)
  LNBOOL HaveSecondsElapsed (LNINT n)
  ```

  These functions return TRUE once every day, *n* minutes or seconds since the add-in started.

  For example, to schedule a task to execute every 2 hours, you would call HaveMinutesElapsed(120), and when this call returned a non-zero value, execute the task.

  Note These functions do not block your add-in. To yield the process control to the Notes server, use one of the following functions.

- Yielding processor control.

  ```
  LNBOOL Idle ()
  LNBOOL Idle (LNINT msecs)
  ```

  These functions yield control of the processor to Notes so that other tasks can execute, and receive control back when the server decides the add-in may proceed.

  If no argument is specified, Notes decides when the add-in should resume. If an argument is specified, Notes suspends the add-in for the specified number of milliseconds; if zero is specified, the function returns immediately.

  A TRUE value is returned from the function call when the Notes server wants the add-in to shut down.

- Checking for task termination conditions.

  ```
  LNBOOL ShouldAddinTerminate ()
  ```

  The function ShouldAddInTerminate will return TRUE when the Notes server wants the add-in to shut down.

  If the task execution of your add-in requires a long time to complete, you may want to call this function periodically to enable the add-in to respond to a termination request sent by the Notes server.

Putting all these functions together, the main loop of a server add-in looks like this:

```
LNNotesSession session;           // Create a session object
LNServerAddin addin;

// Initialize the session
session.init ();

// Get the associated add-in object
session.GetServerAddin("Add In's Name",
                       "<Your description here>",
                       &addin);

while (TRUE)
{
   // check the schedule condition (day, n minutes, n
seconds)
   if (addin.HaveSecondsElapsed (120))
   {
      // perform the operation
   }
   else
      if (addin.Idle (120) == TRUE)
      {
         // Notes server requests you to shut down
         // So, clean up and return
        return;
      }
}
```

Note In general, you may prefer a call to Idle with an explicitly specified wait time, because it may affect the performance of the Notes server.

Status and Log Information of an Add-In Task

When you create a server add-in task, you can provide users with information about the status of the task by specifying messages to be displayed on the server console and stored in the server LOG.NSF log file.

Every add-in task you create has one default status line. If you have a complex task that performs several sub-operations, you may define separate status lines for each of them.

The following functions in the LNServerAddin class enable you to work with the default add-in task status line and the server log:

- Change the task name.

  void SetDefaultStatusLineTaskName (const LNString &name)

- Change the task description.

  void SetDefaultStatusLineText (const LNString &text)

- Add an entry in the server log LOG.NSF.

  void AppendLogMessage (const LNString &message)

  This function is also extremely useful when you need to debug your add-in.

Example: A Server Add-In to Compact Databases

The following C++ program implements a Notes server add-in that runs once every day, and compacts a particular database in the local NOTES directory. You can extend it easily by directory search functions to compact all databases stored on the server once a day.

```
#include <iostream.h>
#include <lncppapi.h>

// Function to compact a database.
// Add here more sophisticated code to compact all
// databases in the local NOTES directory.
void compact_database (LNNotesSession &session)
{
   LNDatabase db;

   session.GetDatabase ("DISCUSAPI.NSF", &db);

   db.Compact ();
}

void main (int argc, char *argv[])
{
  LNNotesSession session;       // Create a session object
  LNServerAddin addin;

  LNSetThrowAllErrors(TRUE);    // Enable exceptions

  try
  {
     // Open a session and get the add-in object
     session.Init();
     session.GetServerAddin("Database Compactor",
                            "Idle",
                            &addin);
     while (TRUE)
     {
       // Start it once a day: returns TRUE
       // at or after 2am in the morning
       if (addin.IsNewDay ())
        {
           // Update the status line
           addin.SetDefaultStatusLineText ("Busy");

           // Call the task execution function
           compact_database(session);

           addin.SetDefaultStatusLineText ("Idle");
        }
        // Here, we will wait always one hour:
        // it doesn't matter whether the task is
        // executed at 2am or at 3am
        if (addin.Idle(3600000) == TRUE)
        {
           addin.AppendLogMessage (
             "Accepting request to shut down");
           return;
        }
     }
  }
  catch(LNSTATUS error)
  {
     // Handle errors:
     // We're on the server, so write it into the log.
     char errorBuf[LNERROR_MESSAGE_LENGTH];

     LNGetErrorMessage(error, errorBuf);
     addin.AppendLogMessage ("An error occurred: ");
     addin.AppendLogMessage (errorBuf);
  }
}
```

Appendix A
Special Notices

This publication is intended to help you develop Notes applications based on Lotus Notes Release 4.5.

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 products, programs, or services may be used. Any functionally equivalent program that does not infringe any IBM 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.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (“vendor”) products in this manual has been supplied by the vendors and IBM assumes no responsibility for its accuracy or completeness. 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 performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

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

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

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

Other trademarks are trademarks of their respective companies.

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

DataLens®
InterNotes
InterNotes Web Publisher
Lotus®
Lotus Notes Reporter
Lotus Notes®
Lotus Notes ViP®
Lotus @SQL®
LotusScript®
Notes
Notes HiTest
Notes ViP®
Notes Mail®
NotesPump
NotesSQL
Notes/FX
Phone Notes®
Phone Notes Mobile Mail
SmartIcons®
Video Notes
Word Pro®

Appendix B
Related Publications

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

International Technical Support Organization Publications

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

- *LotusScript for Visual Basic Programmers*, IBM form number SG24-4856-00, Lotus part number 12498
- *Secrets to Running Lotus Notes: The Decisions No One Tells You How to Make*, IBM form number SG24-4875-00, Lotus part number AA0424
- *Lotus Notes Release 4 In a Multiplatform Environment,* IBM form number SG24-4649-00
- *IBM PC Server and Lotus Notes Integration Guide,* IBM form number SG24-4857-00
- *Lotus Notes on AIX Systems Installation, Customization and Administration,* IBM form number SG24-4694-00
- *Using ADSM to Back Up Lotus Notes,* IBM form number SG24-4534-00

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, IBM form number GG24-3070.

Related Publications

The publications listed in this section may also be of interest:

- *Lotus Notes Network Design*, John Lamb and Peter Lew, McGraw-Hill, ISBN 0-07-036160-6, IBM form number SR-7378-00, http://www.infor.com:53311/cgi/getarec?mgh28004

- *Network Security*, Charlie Kaufman, Radia Perlman and Mike Speciner, Prentice Hall, ISBN 0-13-061466-1, http://www.prenhall.com/013/061465/06146-5.html
- *How to Plan, Develop, and Implement Lotus Notes in Your Organization,* Mike Falkner, John Wiley, ISBN 0-471-12570-9, IBM form number SR23-7262-00, http://www.wiley.com/compbooks/catalog/08/12570-9.html
- *Lotus Notes 4 Administrator's Survival Guide,* Andrew Dahl, Sams Publishing, ISBN 0-672-30844-4, http://www.amazon.com/exec/obidos/ats-query/8244-1222537-793221
- *Deployment and Beyond: The Executive Guide to Implementing Lotus Notes,* Mark Turrell, Imaginatik, fax: +44-171-336-8099, http://www.imaginatik.com
- *ISSC Lotus Notes Operations Cookbook,* IBM Internal publication, ftp://ftp.atlissc.ibm.com/csoffice/docs/lotnotes/opcook.ps

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 URL http://www.redbooks.ibm.com/redbooks.

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 the 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)

  Note The current redbook *Lotus Notes Release 4.5: A Developer's Handbook* is not available as a LIST3820 or in BookManager format.

  - To get lists of redbooks:

    tools sendto wtscpok tools redbooks get redbooks catalog

    tools sendto usdist mkttools mkttools get itsocat txt

    tools sendto usdist mkttools mkttools get listserv package

  - 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 Home Page on the World Wide Web

  http://w3.itso.ibm.com/redbooks/redboooks.html

  Note The current redbook *Lotus Notes Release 4.5: A Developer's Handbook* is also available in HTML format and in Adobe Acrobat format on the World Wide Web. The URL is http://www.lotus.com/devtools.

- 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.

- ITSO4USA 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:

IBM Mail	**Internet**	
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	bookshop at 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
Publications Customer Support
P.O. Box 29554
Raleigh, NC 27626-0570

IBM Publications
144-4th Avenue, S.W.
Calgary, Alberta T2P 3N5
Canada

IBM Direct Services
Sortemosevej 21
DK-3450 Allerod
Denmark

- **Fax** - send orders to:

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

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (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 Home Page	http://ww.redbooks.ibm.com/redbooks
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

  The current redbook *Lotus Notes Release 4.5: A Developer's Handbook* is also available in HTML format and in Adobe Acrobat format on the World Wide Web. The URL is http://www.lotus.com/devtools.

- **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 annouce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

Index

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

❑ Please put me on the mailing list for updated versions of the IBM Redbook Catalog.

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, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.