

Lotus cc:Mail Forms

Getting Started with Sample Forms

Release 6

Copyright

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Lotus Development Corporation, except in the manner described in the documentation.

© Copyright 1994, 1995, 1996 Lotus Development Corporation
55 Cambridge Parkway
Cambridge, MA 02142

All rights reserved. First edition printed 1994. Printed in the United States.

Lotus, 1-2-3, Ami Pro, Word Pro, Lotus Notes, and SmartIcons are registered trademarks and cc:Mail Forms, LotusScript, and Notes/FX are trademarks of Lotus Development Corporation. cc:Mail is a trademark of cc:Mail, Inc., a wholly owned subsidiary of Lotus Development Corporation. Delrina and FormFlow are trademarks of Delrina (Canada) Corporation. Microsoft is a registered trademark and ODBC and Windows are trademarks of Microsoft Corporation. VIM is a registered trademark of Reynolds and Reynolds Company.

Portions developed by Apogee, 12 Alfred Street, Woburn, MA 01801.

Contents

How to Use This Book	ix
Who Should Read This Book	ix
Organization of This Book	x
Using This Book with Other Documentation	x
Help	x
<i>Lotus cc:Mail Forms Designer Application Developer's Guide</i>	x
<i>Lotus cc:Mail Forms Tutorial</i>	xi
<i>LotusScript Language Reference</i>	xi
Conventions Used in This Book	xi
Ad hoc workflow	1
Administrative tasks	2
Printed paper and industry forms	2
Workflow tracking	2
Conditional routing	2
Form Templates	3
Using Template Components	3
Field and non-field layout objects	4
Database links	4
E-mail route	4
Roles database table	5
Tracking database	5
Scripts	5

About New Features in This Release	6
Spell checker	6
Object library	6
Roles Database Editor	7
Picture field enhancements	7
Grouped objects	8
Process Next Form command in Forms Filler	8
Search enhancements	9
Delrina file import	9
Sample form layouts	9

Chapter 2 Starting cc:Mail Forms Designer 11

Installing cc:Mail Forms Designer	11
To install cc:Mail Forms Designer Release 6 under Windows 3.1	11
To install cc:Mail Forms Designer Release 7 under Windows 95	12
Getting Help during installation	12
Starting cc:Mail Forms Designer	12
Switching Between Designer mode and Filler mode	13

Chapter 3	
A First Look	15
The Designer Window	15
The Filler Window	16
Window Components	17
Title bar	17
Control panel	17
Menus	17
SmartIcons	17
Status bar	18
Designer Views	18
Form Layout view	19
Routing view	19
Menu view	20
Script view	20
Quick Menus	21
Layout objects	21
Selecting Layout Objects	22
Selecting field frames, labels, and bodies	22
Selecting frames, labels, and bodies from the properties dialog box	23
Selecting labels and bodies from a Quick menu	23
Selecting bodies in check box and option button fields	23
Properties Dialog Box	24

Chapter 4	
The Properties Dialog Box	25
Displaying the Properties Dialog Box	26
Understanding the Structure of the Properties Dialog Box	26
Selection area	26
Tabs and panels	27
The Help button	27
Moving, Collapsing, and Closing the Properties Dialog Box	28
Chapter 5	
Creating a Phone Message Form	29
Understanding the Procedures for the Phone Message Form	30
Setting Up the Page	31
Creating the Backdrop Rectangles	32
Adding a Combo Box	35
Creating Text Input Fields for the Date and Time	38
Creating the Static Text Field	41
Creating Text Input Fields for the Mr./Ms., Of, and Phone Fields	43
Creating the Check Boxes	45
Creating a Scrolling Text Input Field	49
Creating the Command Buttons	52
Writing the Scripts	54

Chapter 6	
The Object Library	57
Browsing Through the Object Library	58
Copying an Object from the Object library	59
Copying an Object to the Library	60
Changing an Object's Category	61
Chapter 7	
Routing Lists, Roles, and Stops	63
The Routing View	64
Roles Database Table, Editor, and Script	65
Viewing the records in the default roles database table	65
The default roles database table	65
The Roles Database Editor	66
The default script for a role	69
Roles in a Routing List	70
Stops in a Routing List	71
Adding stops to a routing list	71
Pre-processing and post-processing scripts for stops	74
Chapter 8	
Linking Layout Objects to Database Fields	75
Creating Database Links	75
Working with Database Link Windows	76
Creating a database link window	76
Linking a field layout object to a database field	77
Hiding a database link window	79
Deleting a database link	79
Modifying database fields from a database link window	79
Specifying properties for a database link window	80
Linking a Pick List to a Database Table	81
Determining When Forms Filler Queries a Database	83
Using automatic query	83
Running a query from the Forms Filler	83
Using a script to query a database table	83
Finding More Information on Databases	84
Chapter 9	
Setting Up a Tracking Database	87
Specifying the Information to Track	87
Enabling Tracking for the Stops	89
Advanced Uses of Tracking Databases	90
Chapter 10	
Customizing Menus	91
Understanding Menu View	92
Adding a Menu Command	93
Removing a Menu Command	95
Removing a Main Menu Command	96
Reinstating a Standard Command	96

Chapter 11
Using cc:Mail Forms Designer and Notes/FX 99

About Notes/FX	99
An Example of Notes/FX	100

Chapter 12
Designing Forms 105

Guidelines for Developing Form Templates	105
Starting the form template	106
Designing new templates	106
Testing the form	110
Guidelines for Creating a Database Table	110
Distributing a Form and Its Template	111

Chapter 13
Working with the Sample Forms 113

Understanding the Sample Post Office	114
Opening a Sample Form Template	114
Viewing Scripts	115
Displaying a Form's Routing List	116
Displaying Database Link Windows	116
Switching Between Designer Mode and Filler Mode	117
Closing a Sample Template	117
Getting Help for Running the Form Applications	118

Chapter 14
A Template Developer's Look at the Sample Forms . . . 119

Phone Message Form	120
Template file	121
Feature highlights	121
Creating a three-dimensional effect	121
Date- and time-stamping a new form	121
Creating a combo box list with a pick list and a script	122
Establishing a DDE link to e-mail	122
Clearing all data in a form	123
Application for Employment Form	124
Template files	125
Feature highlights	125
Using command buttons for navigation	126
Storing information in database tables	126
Using one form to open another form	127
Transferring information from one form to another	127
Printing a form with a button	128
Executing a menu command with a button	128
Using a roles database	129

New Employee Form	130	Quotation Record Form	145
Template file	131	Template file	145
Feature highlights	131	Feature highlights	146
Reading and writing data to and from database tables	131	Specifying a criterion to query a database	146
Using combo boxes in tables	132	Integrating Word Pro and cc:Mail in a form application	146
Displaying fields at specific stops	132	Purchase Order Form	149
Weekly Time Sheet Form	133	Template file	149
Template file	133	Feature highlights	150
Feature highlights	134	Entering default values in fields	150
Totaling table and row columns	134	Creating a DDE link with Lotus 1-2-3	150
Tracking a form's progress through a route	134	Conditional routing of a form	151
Displaying a field at a specific stop	135	Enabling fields at specific stops	151
Expense Report Form	136	Receiving Summary Form	152
Template file	137	Template file	153
Feature highlights	137	Feature highlights	153
Querying a database table automatically	137	Exchanging field data with Lotus Notes	153
Totaling and subtotaling table rows and columns	137	Using a hidden field to determine a form's status	154
Request for Quotation Form	138	Initializing a form	155
Template file	139	Lead Response Form	156
Feature highlights	139	Template file	156
Linking a pick list to a database	139	Feature highlights	157
Moving information from one form list to another list	140	Using a script to connect to a database	157
Using a script to route a form	142	Using buttons to display database records	157
Writing information to a database	143	Establishing a DDE link to e-mail	158
Hiding fields at a specific stop	144		

Trip Reservation Form	159
Template files	160
Feature highlights	160
Transferring information from one form to another	160
Creating a custom menu command	162
Displaying fields based on the state of another field	162
Routing a form to more than one recipient at a stop	163
Roles Database Editor	164
Template file	165
Feature highlights	165
Connecting to a database table and displaying the first record	165
Displaying message boxes	167
Displaying a record counter	168
Clearing displayed data	168
Adding and updating database records	168
Deleting the current record	170

How to Use This Book

For more than a decade, Lotus® software has been the world's standard for desktop, communications, and development applications. cc:Mail Forms Designer is an advanced forms tool for forms management.

cc:Mail Forms Designer is a development tool designed for creating and implementing forms automation across an organization. You use cc:Mail Forms Designer to create a form template for a specific task and to provide your users with a graphical interface to view, organize, and track business process information.

Lotus cc:Mail Forms Designer Getting Started with Sample Forms introduces you to the power and flexibility of cc:Mail Forms Designer. This book includes information on the following activities:

- Using cc:Mail Forms Designer to solve workflow problems
- Working with features such as scripting, routing, and database links
- Implementing the features included with the sample forms

Who Should Read This Book

Lotus cc:Mail Getting Started with Sample Forms is for developers who already have a basic familiarity with creating forms, and who want to learn how to automate business processes in their own work environment.

As you read the chapters in this book, you can refer to the sample forms to see how they use cc:Mail Forms Designer features. You can run the sample forms, filling in the form as it proceeds from mail stop to mail stop. Or, you can work with the form templates as a developer, viewing database links, scripts, routing lists, and field properties.

Organization of This Book

Lotus cc:Mail Getting Started with Sample Forms is organized into the following chapters:

- Chapters 1 and 2 provide an overview of cc:Mail Forms Designer, descriptions of the features new to this release, and instructions for installing the software.
- Chapters 3 through 10 show how to create a template and work with template components such as routing lists, database links, and customized menus.
- Chapter 11 explains how to exchange data between cc:Mail Forms Designer and a Lotus Notes database.
- Chapter 12 offers guidelines for developing templates.
- Chapters 13 and 14 take you on a developer's tour of the sample forms included with cc:Mail Forms Designer. These chapters describe the sample templates and provide information on how to implement the features included with the templates.

Using This Book with Other Documentation

Help

Lotus cc:Mail Getting Started with Sample Forms and online Help complement each other. *Lotus cc:Mail Getting Started with Sample Forms* covers basic concepts and provides step-by-step descriptions of creating templates and working with template components, along with illustrations and examples. This book also describes the key features implemented in the sample form templates. Online Help documents every cc:Mail Forms Designer feature and includes procedures for using each of the sample forms.

Lotus cc:Mail Forms Designer Application Developer's Guide

Lotus cc:Mail Forms Designer Application Developer's Guide is for form application developers. It contains basic and advanced information about using cc:Mail Forms Designer. If you're already familiar with other forms software, reading *Lotus cc:Mail Forms Designer Application Developer's Guide* is a good way to learn how to use this product's commands and features.

Lotus cc:Mail Forms Tutorial

The tutorial introduces cc:Mail Forms Designer to the novice user. Start with *Lotus cc:Mail Forms Designer Tutorial* if you are new to form development and want procedures that walk you through the process of creating objects and other template components. As you become familiar with the concepts presented in the tutorial, you'll be ready to move on to *Lotus cc:Mail Forms Designer Application Developer's Guide*.

LotusScript Language Reference

LotusScript™ is a BASIC-compatible programming language that you use to automate your work in Lotus products. *LotusScript Language Reference* provides information on the commands and features available across Lotus products.

For information on the LotusScript commands and features specific to cc:Mail Forms Designer, refer to Chapters 8, 9, and 10, and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide*.

Conventions Used in This Book

Lotus cc:Mail Getting Started with Sample Forms uses the following conventions:



- SmartIcons® next to a procedure step indicate that you can click the icon instead of choosing the specified command.
- Menu commands are separated by a - (hyphen); for example, File - Open Form Template.
- Function keys appear in small capitals and are identified by the cc:Mail Forms Designer key name; for example, **F1 (HELP)**.
- **Bold** new terms and characters that you must type.
- **See** cross references other topics, information in other books, or Help.
- **Note** introduces additional information about a command or procedure.
- **Tip** tells you about a shortcut or gives you advice on how to use one feature in combination with another.
- **Caution** describes possible negative consequences of performing an action, including loss of data.

Chapter 1

Introducing cc:Mail Forms Designer

cc:Mail Forms Designer is a development tool for creating and implementing forms automation across Lotus cc:Mail™. You use cc:Mail Forms Designer to create a form or a set of related forms for a specific task. This capability provides users and their managers with a structured, graphical means with which to view, organize, and track business process information.

As the form developer, you define the actions taken when a user enters data, clicks a button, or chooses a menu command. You also define the properties of all fields, including pop-up lists, buttons, text input fields, and database links.

To help you gain an overall understanding of cc:Mail Forms Designer, this chapter covers the following tasks:

- Solving workflow application problems
- Defining form templates and documents
- Using template components
- Using new features of cc:Mail Forms Designer

Solving Workflow Application Problems

You can create a form application for any administrative process that involves gathering information from one or more users or sources and that uses e-mail to send the information to other destinations. The following sections describe some sample applications.

Ad hoc workflow

The most common type of workflow is a form that is sent around an organization. The easiest way to accomplish this routing is to take advantage of your existing e-mail systems. With cc:Mail Forms Designer you have access to the full capability of your e-mail system, and you can add a routing list to a form to automate the mailing process.

Administrative tasks

You can create new forms from scratch, or modify existing forms as the basis for new forms. You can use forms to process new hire information, purchase requisitions, expense reports, time sheets, requests for literature, and messages, to name a few.

Printed paper and industry forms

Rather than having to reinvent industry-standard forms for electronic form design and development, you can scan an existing paper form and trace a new form over it. The result is an electronic form that is familiar to your users and that can easily be printed onto its corresponding paper form. This is particularly useful for industry-specific forms, such as government and insurance forms.

Workflow tracking

Monitoring a form's progress as it moves across an organization often needs to be part of the application's workflow rather than a separate application. cc:Mail Forms Designer and Forms Filler in cc:Mail can do the following tasks:

- Write information from a form to a tracking database after the form reaches specific stops or when a user performs a specific action
- Mail a copy of a form to a stop on the routing list or to any mailbox outside the routing list

Conditional routing

Using the LotusScript language, you can define conditional routing. For example, a purchase requisition of more than \$10,000 may require a manager's approval and a vice president's approval. After receiving the manager's approval, the routing script can check the amount of the requisition and determine whether the form must go to the vice president or go directly to purchasing.

Form Templates

You, as the developer, design a *form template*. The user, however, enters information into the finished *form*. Here is the distinction between these two terms.

- The **form template** defines the content and layout of a finished form. A template can include text input fields, bitmaps, database links, formulas, electronic signatures, and route addresses.
- The **form** is the finished product; the data-entry interface that an end-user completes. In a form, users can enter data, create pop-up notes, and write annotations. Users cannot change the form template itself; they cannot move checkboxes or delete fields, for example.

The developer creates the form template in Designer mode, then tests it in Filler mode. Users work with the form in the Forms Filler, which is included in Lotus cc:Mail, to perform data entry and other tasks specified by the form template.

Using Template Components

As you build a form template, you add the following components to implement the tasks you want the user to perform:

- Field and non-field objects
- Database links
- E-mail route
- Roles database table
- Tracking database
- Scripts

The following sections provide an overview of these components.



See Chapters 3 - 14 of this book, Help, and *Lotus cc:Mail Forms Designer Application Developer's Guide* for detailed information on how to use these components.

Field and non-field layout objects

A layout object is anything you place in a form template, such as command buttons, rectangles, text input fields, check boxes, lines, circles, and bitmaps. The two types of objects are field and non-field.

A field object can accept input and can display data. For example, a user can select a check box, choose an option from a list box, type an entry in a text input field, or click a command button. The field's data type determines the type of data the field can contain: text, numbers, time, date, or electronic signature.

A field object can also have properties that trigger events. For example, when a user clicks a command button, another section of a form can appear, information can be written to a database, or the form can be sent to the next stop in the routing list. You define an event using the LotusScript language to write a script for a specific field.

A non-field object neither accepts input nor displays data. For example, rectangles and circles cannot display data, nor can a user enter or select information in these objects. Non-field objects are primarily used as design elements, to make the form easier to read and use.

Database links

You can link a field on a form template to any Open Database Connectivity (ODBC™) database. This means a form can read and write information to a database table. A form template can have links to as many different database tables as you need.

E-mail route

You have the option of defining the route that a form follows to travel through an organization. The route is defined by a list of stops that have either e-mail addresses or roles. Because the routing information is part of the form, you can send a form across multiple mail systems.

Routing works with Vendor Independent Messaging (VIM®)-compliant mail systems such as Lotus cc:Mail. You can use cc:Mail Forms Designer to directly access cc:Mail's address book to send and receive forms.

You can send forms to other users by attaching the form to a mail message or automatically sending a form to a mail-enabled application.

Roles database table

A role is a representation of a job title, such as manager, vice president, and human resources manager. A roles database table defines who is filling the role. For example, for employee B, the roles database table would list the e-mail address of employee B's manager, vice president, and human resources manager. cc:Mail Forms Designer uses the roles database table to reconcile a role in a routing list with an e-mail address. Roles are defined using any ODBC database.

Tracking database

A tracking database monitors a form's progress through an organization. At each stop, cc:Mail Forms Designer can write information to the database table, such as a form identifier, date and time stamp, who sent the form, and who received it. As the developer, you can specify that cc:Mail Forms Designer writes information from any of the form fields to the tracking database. You can use any ODBC database source to create the tracking database table.

Scripts

You use the LotusScript language to write underlying scripts that control events in a form, such as the following examples:

- A pre-processing script can determine properties for a stop in a routing list. For example, you may want to hide fields that contain confidential information for some of the stops on a routing list.
- A script for a menu command can specify what happens when a user chooses that command. For example, you could customize the action of File - Save to save and print a form.
- A script for a command button in a form can determine what action occurs when a user clicks the button. For example, the action could be going to another page in a form or writing information to a database.

About New Features in This Release

The following features are new to this release of cc:Mail Forms Designer.

- Spell checker
- Object library
- Roles Database Editor
- Picture field enhancements
- Grouped objects
- Process Next Form command in Forms Filler
- Search enhancements
- Delrina™ file import
- Sample form layouts

The following sections give an overview of each new feature.

Spell checker

The spell checker reviews the spelling of all text in field labels, command buttons, check boxes, option buttons, and static text fields. When the spell checker finds a word that is not in the dictionary, you can replace, edit, skip, or add the word to the dictionary.



The spell checker is available in Filler mode and in the Form Layout view of Designer mode. You start the spell checker by clicking its icon or choosing Edit - Spell Check.



See Chapter 3 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and search on “spell check” in Help.

Object library

The object library provides you with pre-scripted objects for frequently used layout objects and frequently performed tasks. For example, you can copy the following layout objects from the object library into a form template:

- Buttons that go to the next page, send a form to the next stop in the routing list, or add an electronic approval signature to a form
- Text input fields that amortize a loan or determine the day of the week
- Signature fields that fill in the date and time when the form is signed

The object library also has sets of objects that together implement a feature, such as the following features:

- Adding scroll bars to a table
- Retrieving the user's name and mail system from VIM.DLL
- Adding an ODBC data source at runtime

After you copy an object from the library into a form template, you can use the object as is, or modify the object and/or its script. You can also create and script your own layout objects, and add them to the object library.

The object library is available in the Form Layout view of Designer mode. You access the object library by clicking its icon or choosing File - Object Library.

See Chapter 6 in this book, Chapter 4 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and search on "object library" in Help.

Roles Database Editor

The Roles Database Editor is a sample form that is used by developers to browse and update the records in a roles database table. You enter the role name, from name, and to name, and click the appropriate button to have the Roles Database Editor add, update, or remove the corresponding record from the database table. You can use the Roles Database Editor "as is" to add your own roles to ROLES.DB (the default roles database table), or you can change the settings in the Roles Database Editor's template to modify a different roles database table.

As with the other sample forms, you can view the form template (ROLES.LTM) for the Roles Database Editor in Designer mode, and use the sample form in Forms Filler window to modify a database table.

See Chapters 7 and 14 in this book, and search on "roles database" in Help.

Picture field enhancements

Several new features for picture fields let you do the following tasks:

- Resize an image by dragging its selection handles
- Maintain aspect ratio as you resize an image
- Crop an image by typing the amount to crop (for example, 1 inch from the top or .5 inch from the left)
- Change the image in a picture field by typing the file name of the new image

- Refer to a picture object in a script (such as writing a script that changes which image appears in a picture field when the form is opened at a specific stop)

Tip In the script, use the Update method to change the source image for the picture field.

The new features for picture fields are in the properties dialog box for a picture field object (in the Form Layout view of Designer mode). The cropping feature is also available in the Forms Filler, for picture fields that a user adds to a form.



See Chapter 4 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and click the Help button in the properties dialog box for a picture field.

Grouped objects

You can now group layout objects as you create a form template. This means you can select two or more layout objects and specify that you want them grouped. You can then select the group as a single object, and handle it as a single object. For example, you can move the group to another location without losing the relative placement of the individual objects to each other. You can also drag a selection handle to size all objects in the group. The properties dialog box for a group displays the common settings that you can change for the selected group.



You group layout objects in Form Layout view. Select the layout objects to group and then either click the Group icon or choose Edit - Group. To ungroup the objects, select the group and then either click the Ungroup icon or choose Edit - Ungroup.

See Chapter 4 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and search on "group" in Help.

Process Next Form command in Forms Filler

Forms Filler (in Lotus cc:Mail) has a new command that lets users open additional forms attached to a mail message without relaunching Forms Filler. For example, a supervisor may have several time sheets to sign. The supervisor opens the e-mail message with the attached time sheet and launches Forms Filler to sign the time sheet. Once in Forms Filler, the supervisor can choose Process Next Form to open the next time sheet. It's no longer necessary to exit the first time sheet and then restart Forms Filler to sign the next time sheet.



This command is available in Forms Filler only. The Forms Filler user can click the Process Next Form icon or choose File - Process Next Form. If the user prefers using the icon, it must first be added to the SmartIcons palette.

See File topic in Help, by searching on "File commands."

Search enhancements

A form saved without its template is considerably smaller than one saved with the template. However, you must plan the distribution of a form saved without a template. You must make sure that the Forms Filler can find the corresponding template when a user opens a form.

Forms Filler can search Notes databases and cc:Mail bulletin boards for a form's template. The Template Search Path in the Forms Designer Preferences dialog box lets you widen the search scope for a corresponding template. For Forms Filler to search a Notes database or a cc:Mail bulletin board, the Notes or cc:Mail application does not have to be running; however, the Notes server must be running.

You must make sure that the name of the Notes database or the cc:Mail bulletin board is added to the Template Search Path in the Forms Filler. As the developer, you would either have your Forms Filler users add this information, or you would add it as part of setting up your Forms Filler users. In addition, you must provide cross-certification of Notes IDs on the Notes server to allow Notes access. Refer to your Notes documentation for more information.



See Chapter 11 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and "Distribute a Form" in the How Do I section of Help.

Delrina file import

You can import a Delrina FormFlow™ 1.0 .FRP file into cc:Mail Forms Designer, thus converting the Delrina file to a cc:Mail Forms Designer form template. You can import locked FormFlow file and objects.

To import a Delrina FormFlow 1.0 .FRP file, choose File - Open Form Template and then select the .FRP file type.



See the Using Import File Filters topic in Help, by searching on "importing."

Sample form layouts

cc:Mail Forms Designer includes a collection of sample form templates that are standardized business formats, to help you in designing forms.

These sample templates are automatically installed in the C:\DESIGNER\WORK\LAYOUTS directory on your hard disk. If you are using Notes, they are included in a Notes database called LAYOUTS.NSF in the document/view called SAMPLES.

The sample templates are grouped into the following categories.

- Government templates include a travel voucher, an order for supplies and services, requests and authorizations for travel and leave, and a personnel action form template.
- Administration templates include a bill of lading, a packing list order, a receipt of goods, an inventory record, an invoice, credit and debit memos, and a tuition advance and reimbursement form template.
- Insurance templates include a variety of standard forms for the insurance industry, including an accident report, individual health insurance claim form, and health care expense reimbursement request form template.
- MIS templates include a user ID request, as well as other standard request layouts for management information services.

Chapter 2

Starting cc:Mail Forms Designer

This chapter has the information you need to set up cc:Mail Forms Designer. It covers the following tasks:

- Installing cc:Mail Forms Designer
- Starting cc:Mail Forms Designer
- Switching between Designer mode and Filler mode

Installing cc:Mail Forms Designer

Before you can use the sample forms, you must use the Install program to transfer the program files to your hard disk. To install the program for the first time or to add optional features later, use the installation disks in your cc:Mail Forms Designer package.

To install cc:Mail Forms Designer Release 6 under Windows 3.1

The directions that follow assume you're starting Install from a high-density A drive. If you start Install from a different drive, substitute the letter of that drive in step 4.

1. Insert the cc:Mail Forms Designer Install Disk 1 in drive A.
2. Open the Windows Program Manager or Windows Explorer.
3. Choose File - Run.
4. Type **a:install**.
5. Click OK.

A series of dialog boxes appear, asking for information about what to install and how to install it.

To install cc:Mail Forms Designer Release 7 under Windows 95

The directions that follow assume you're starting Install from a high-density A drive. If you start Install from a different drive, substitute the letter of that drive in step 1.

1. Insert the Install Disk 1 in drive A.
2. Press the Start button to bring up the Windows System menu.
3. Select Run.
4. In the Open: text box, type **a:install**.
5. Click OK.

An introductory screen appears.

Getting Help during installation

During installation, you can get explanations of the options in each dialog box.

1. Click the Help icon or the Help button.
2. To display another Help topic, click the appropriate Help button.
3. Choose File - Exit to close the Help window and return to the Install dialog box.

Starting cc:Mail Forms Designer

Once you have installed cc:Mail Forms Designer, you are ready to start the program.

1. Open the Windows Program Manager or Windows Explorer.
2. Open the Lotus Applications window (or the group window that contains Forms Designer).
3. Double-click the Forms Designer application icon.

The program title screen appears briefly, and then the Designer window opens.



Switching Between Designer mode and Filler mode

As you work, you will frequently switch between Designer mode and Filler mode. You can do this quickly with SmartIcons.

- Click the Filler icon to go from Designer mode to Filler mode.
- Click the Designer icon to go from Filler mode to Designer mode.



The Filler window is a test environment where you can test a form, to make sure it will work correctly for your users. Any information or data that you enter in the Filler is not saved when you return to Designer mode. If you want to save the data, use the File - Save command in Filler mode to create a form.

Chapter 3

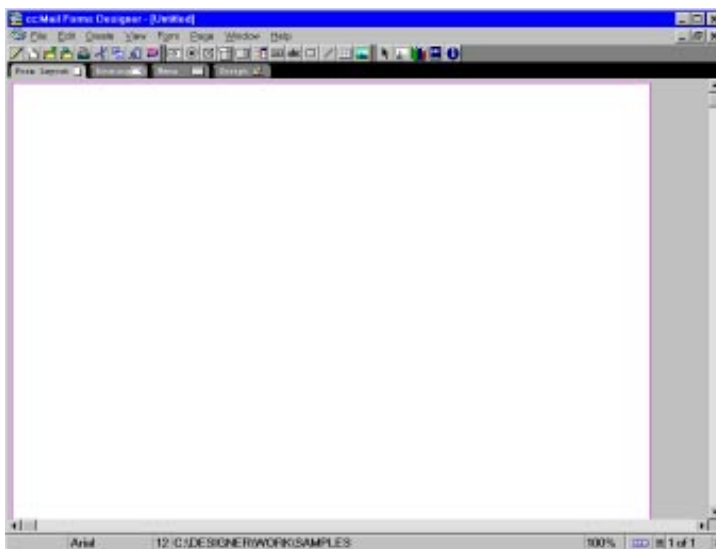
A First Look

cc:Mail Forms Designer shares the same intuitive and easy-to-use design with other cc:Mail Windows applications. This chapter explores the following key features of the cc:Mail Forms Designer interface:

- Designer and Filler windows
- Designer window components
- Views in the Designer window
- Quick menus
- Layout objects
- Object selection
- Properties dialog box

The Designer Window

The Designer window is a complete Windows-based graphical design environment. It contains a wide range of layout and graphical tools for creating form templates.



Using the tools in the Designer window (such as text format control, rulers, grids, alignment, paragraph indentation, and justification) you can create exact replicas of existing paper forms.

The Designer's graphical tools (such as command buttons, check boxes, option buttons, and list boxes) let you create graphical screens that are easy to read and fill out.

See Chapter 2 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more details on the cc:Mail Forms Designer environment.

The Filler Window

The Filler window displays the same interface as the Forms Filler that users see in cc:Mail. As the form developer, you can test the design of your form templates in the Filler window by entering and editing data, performing queries, and printing forms.

Menus available to user are defined at design time by the developer

The screenshot shows the Lotus cc:Mail Forms Designer application window titled "cc:Mail Forms Designer (PO1156)". The menu bar includes File, Edit, Create, View, Form, Check, Window, and Help. The toolbar contains icons for various form design actions. The main form area displays a "Purchase Order" template with the following sections:

- Header:** Includes fields for Date, Customer, P.O. Number, Supplier Number, R.F.Q. Number, and Buyer's Name.
- Vendor:** Includes fields for Vendor Name, Company Name, Address, Vendor's City, State, Zip, Vendor's Phone, and Address (Home/Dept.).
- Shipping & Payment:** Includes fields for Ship Code, Delivery Date, P.O.B., Ship Center, Ship Via, Payment, and Payment Terms.
- Ship To:** Includes fields for Company Name, Address, City, State, and Zip.
- Bill To:** Includes a checkbox for "Same as Ship To" and fields for Company Name, Address, City, State, and Zip.

The status bar at the bottom indicates "100% 2011/1/11 11:11".

See Chapter 2 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for details about the Filler window environment.

Window Components

Both the Designer mode and Filler mode windows use the same window components.

Title bar

The title bar includes the program name, the File name, the Minimize button, the Maximize button, and the Close button.



Control panel

You can use the control panel to display the main menu.



Menus

The menus are streamlined and context-sensitive. Contents of the menus vary; offering only the commands you need for the current object.

For example, when you create or select a text field, the Field menu appears.

When you create a text field, the menu displays the Field command



SmartIcons



SmartIcons are shortcuts for many tasks.

For example, you can align several objects by clicking an icon instead of choosing Edit-Align Right.

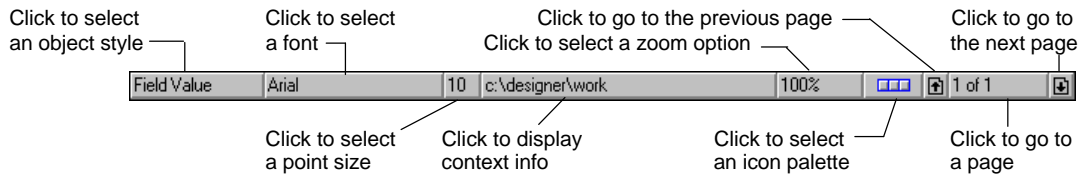
cc:Mail Forms Designer comes with default sets of SmartIcons called palettes. You can move, hide, and customize palettes to contain only the icons you want.

See Appendix A in *Lotus cc:Mail Forms Designer Application Developer's Guide*.



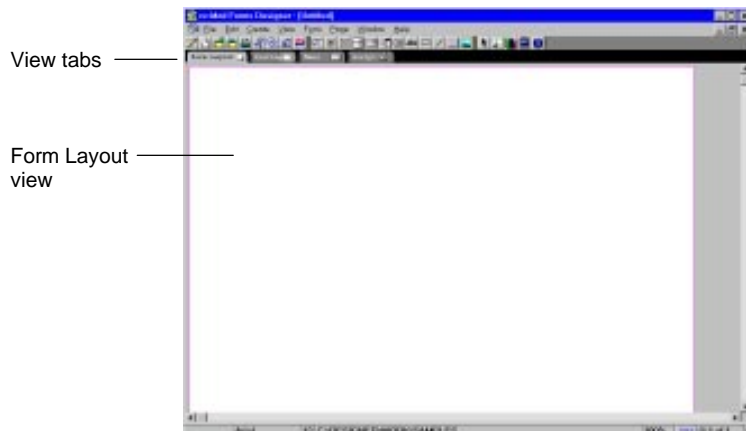
Status bar

The status bar at the bottom of the screen displays settings for the current layout object(s). You can click the status bar to change the styles, fonts, and more.



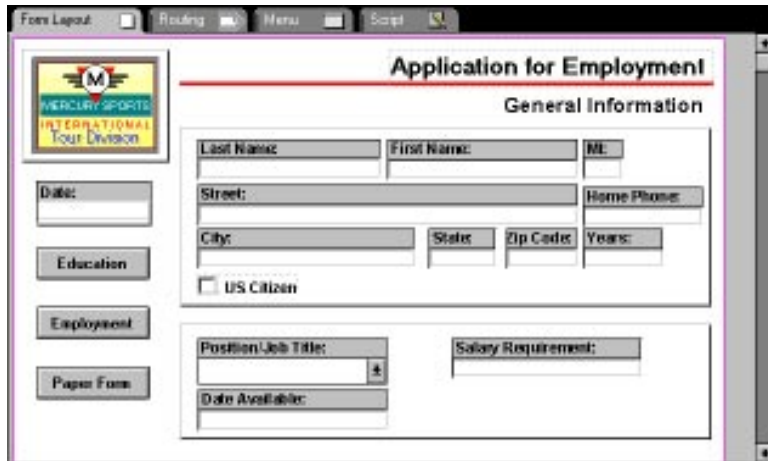
Designer Views

In the Designer window, you lay out and customize form templates in several different views. To switch from one view to another, either click a tab or use the View commands. The four views in the Designer window are Form Layout, Routing, Menu, and Script.



Form Layout view

In Form Layout view, you design the appearance of the form by creating text fields, list boxes, command buttons, and other layout objects.



The screenshot shows the 'Form Layout' view of a software application. The title bar includes tabs for 'Form Layout', 'Routing', 'Menu', and 'Script'. The main window displays a form titled 'Application for Employment' with a sub-header 'General Information'. On the left, there is a logo for 'MERCURY SPORTS INTERNATIONAL Four Division' and a vertical stack of buttons: 'Date:', 'Education', 'Employment', and 'Paper Form'. The form fields include: 'Last Name', 'First Name', 'ME', 'Street', 'Home Phone', 'City', 'State', 'Zip Code', 'Years', and a checkbox for 'US Citizen'. Below these, there are fields for 'Position/Job Title', 'Salary Requirement', and 'Date Available'.

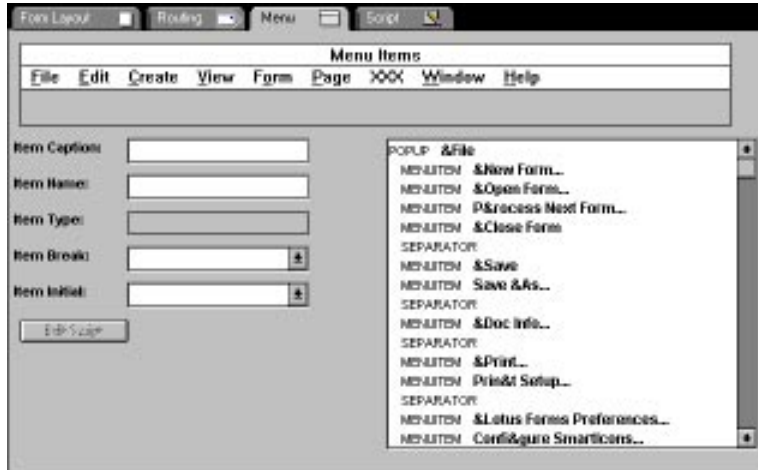
Routing view

In Routing view, you create workflow routing lists that route forms to individuals by name or by role.



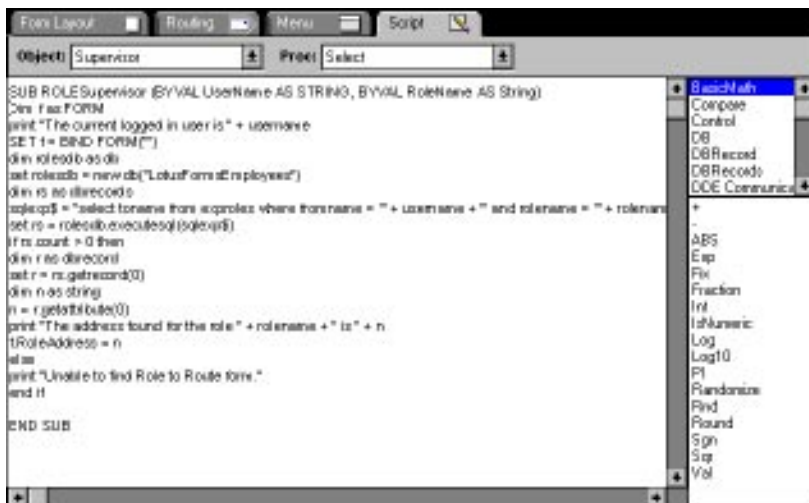
Menu view

In Menu view, you create custom menus and commands for a form.



Script view

In Script view, you create scripts to control the actions triggered when a user clicks an object, to specify conditional routing, and to control other elements of information flow in a form.



Quick Menus

Pressing the right mouse button is a quick way to reach a menu of commands tailored to the current layout object.



The specific options on each Quick menu vary, depending on the layout object you have selected.

Layout objects

There are two types of layout objects: field and non-field. A field object can accept input data and can display data. Text input fields, option buttons, check boxes, combo boxes, and list boxes are all field objects.

Non-field objects are primarily design elements. Tables, command buttons, static text, pictures, rectangles, ellipses, and lines are all non-field objects.

A field object has three parts: field Frame, field Label, and field Body. The next illustration shows the parts of a text input field.



All three parts can have unique settings and properties. For example, a text input field can have one font for the field Label and a different font for the user input in the field Body. The field Frame can be a variety of colors or line styles, or it can be transparent.

See Chapter 4 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information on form objects.

Selecting Layout Objects

You can use the following techniques to select layout objects.

- To select one object, click it.
- To select two or more objects, click the first object, then **SHIFT**+click each additional object. **SHIFT**+click also deselects an individual object.
- Drag a box around the object, or objects, to select.
- Use the Edit - Select All command to select all objects or the same parts of all objects.

When you click the right mouse button over a layout object to display a Quick menu, you also select the object.

Once you start building a form template, the easiest way to select the page or form is from the properties dialog box for the page or form.

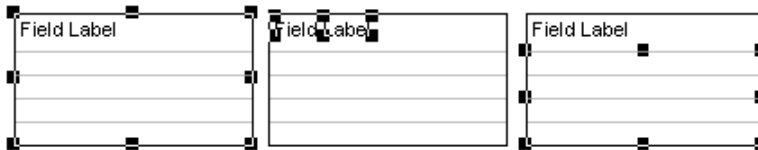
Tip You can also select several objects, then group them as a single object.

See “Work with Layout Objects” in the How Do I section of Help.



Selecting field frames, labels, and bodies

You can select a field object's frame, label, and/or body. The following illustrations show (left to right) the selected frame, label, and body of a text input field.



The Select Field Frame First setting (in the Preferences dialog box) determines which field part is selected when you click a field. cc:Mail Forms Designer selects either the object part the pointer is on or the field frame regardless of the pointer position.

- If the Select File Frame First check box is selected (the default) and you click an object, cc:Mail Forms Designer selects the field frame first. For example, if you point to an object's label and click, cc:Mail Forms Designer selects the object's frame. Click a second time to select the object's label. (These are two separate clicks, not a double-click.)
- If the Select File Frame First check box is not selected, the field part you're pointing to is selected. For example, if you point to an object's label and click, cc:Mail Forms Designer selects the label. To select the frame, point to the frame and then click.

Selecting frames, labels, and bodies from the properties dialog box

Once any part of a field object is selected, you can select a different part with the mouse or from the properties dialog box. This is convenient because you can choose all the settings for a field's label, frame, and body in one properties dialog box.

1. Double-click the field object.
2. In the properties dialog box, under Properties for, choose the field part (frame, label, or body) you want to select from the second drop-down list box.

Selecting labels and bodies from a Quick menu

When the Select Field Frame First check box is selected, a Quick menu useful for selecting a field object's label or body.

1. Place the mouse pointer on the field object and click the right mouse button.

The field frame is selected and the Quick menu appears.

2. Choose one of the following:
 - Select Field Label Only
 - Select Field Body Only

Selecting bodies in check box and option button fields

In check box and option button fields, the body has the following two components: button and caption.

- The button is the box or button that the user clicks in the Filler window.
- The caption is the text that identifies the button.

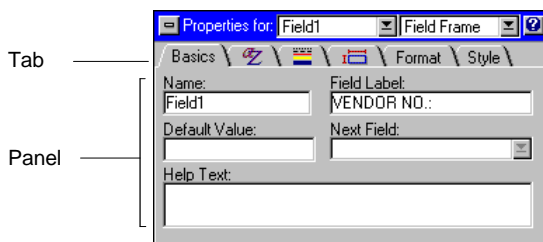
You can select just the button, just the caption, or both.

Tip To simplify selecting these body components, choose File - cc:Mail Forms Designer Preferences and deselect the Select Field Frame First setting. Then click the body component you want. Use **SHIFT**+click to select additional buttons and/or captions.

If Select Field Frame First is selected and you have selected a button or caption and now want to restore the body selection to both the button and caption, use the properties dialog box. Double-click the object and then select the Field Body in the properties dialog box. cc:Mail Forms Designer selects both the button and the caption.

Properties Dialog Box

You use the properties dialog box to select the options you want for a layout object or for part of an object. To display a properties dialog box, double-click an object.

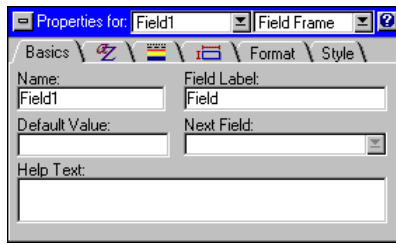


See Chapter 4 in this book, and Chapter 2 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for details on using the properties dialog box.

Chapter 4

The Properties Dialog Box

All objects, fields, and pages in a form template have design properties. For example, objects can have line widths, text fonts, and background colors. Page properties include the page size and color. There are also properties specific to a form. Form properties include the automatic sizing of the window when a form is opened, and the presence or absence of scroll bars. The cc:Mail Forms Designer properties dialog box gives you quick access to all these settings.



Because cc:Mail Forms Designer uses one properties dialog box instead of many dialog boxes, the properties dialog box can stay open as you work. The settings in the box automatically change according to the currently selected object. For example, if you double-click an object, cc:Mail Forms Designer selects the object and opens the properties dialog box displaying the settings for the selected object. If you then click another object, the settings change to show the settings for that object.

This chapter provides the following information about the properties dialog box:

- Structure of the properties dialog box
- Instructions for moving, collapsing, and closing the properties dialog box

Displaying the Properties Dialog Box

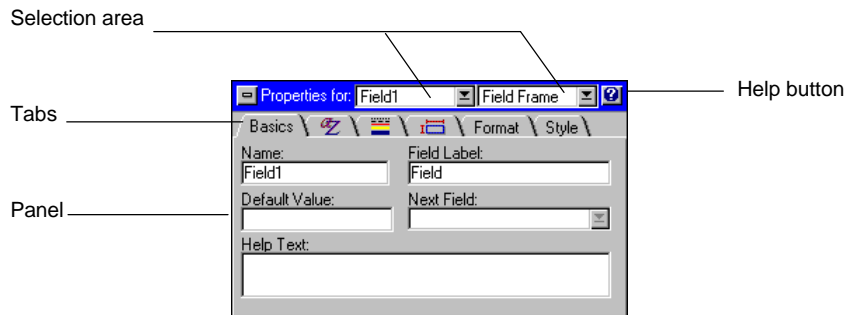
To display the properties dialog box, perform one of the following actions:



- Double-click an object.
- Click the Properties SmartIcon.
- Click the right mouse button on an object, and then choose Properties from the Quick menu.
- Select an object (or part of a field object), and then choose Properties from the object's menu.

Understanding the Structure of the Properties Dialog Box

The primary components of the properties dialog box are the selection area, the tabs and panels, and the Help button.



Selection area

You use the properties for list boxes to change the current selection. Depending on the type of object selected, there are one or two list boxes.

The choices in the first list box are the currently selected object, Page, or Form. When the current selection is the page, the list selections are Page and Form. When the current selection is the form, Form is the only option listed.

By choosing an option from this list box, you change the selection to the page or form. The properties dialog box settings change to those for the page or form. The page settings are specific to the current or all form pages, and the form settings are specific to the entire form.

A field object (such as a check box), has a second list box. The choices in the second list box are Field Frame, Field Label, and Field Body.

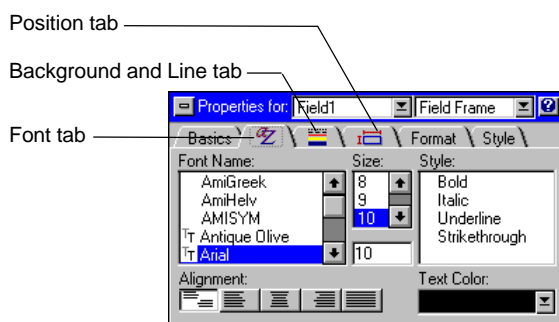
You use this list box to change which part of the object is selected.

When you choose an option from this list box, the settings in the properties dialog box change to reflect the currently selected field part.

See “Layout Objects” in Chapter 3, and Chapter 4 in *Lotus ccMail Forms Designer Application Developer's Guide*.

Tabs and panels

The settings in a properties dialog box are organized by type of setting. Each type of setting has a tab and a panel. The following illustration shows the Font panel for a text input field.



Each panel has a tab. To move to another panel, you click its tab. For the frame of a text input field, the properties dialog box has the following panels: Basics, Font, Background and Line, Position, Format, and Style.

The types of settings are not the same for all objects. For example, the settings for a check box field are different and more extensive than the settings for a rectangle. The properties dialog box displays different settings, tabs, and panels for different object types, the page, and the form.

The Help button



When you need help with the settings on a panel, either press **F1 (HELP)** or click the Help button in the upper-right corner of the properties dialog box. A Help topic specific to the type of object selected appears. If the object is a field object, a Help topic specific to the part of the object you selected appears. For example, if you select the label of a text input field and click the Help button in the properties dialog box, cc:Mail Forms Designer displays a Help topic on the settings for the label of a text input field.

Moving, Collapsing, and Closing the Properties Dialog Box

You can move, collapse, and close the properties dialog box as you work in cc:Mail Forms Designer.

1. To move the properties dialog box, drag its title bar.
Tip The easiest place to grab the title bar is over the text “Properties for.”
2. To collapse the properties dialog box, double-click the title bar.
Only the properties dialog box title bar and tabs remain visible.
3. Double-click the title bar again to expand the properties dialog box.
4. To close the properties dialog box, double-click the control-menu box in the upper-left corner of the properties dialog box.

Chapter 5

Creating a Phone Message Form

This chapter walks you through the process of creating a phone message form.

This phone message form is based on the MESSAGE.LTM template used for the Phone Message sample form application. You may want to keystroke the procedures and create a new form template, or retrieve the existing template and view it as you read through the procedures.

The sections in this chapter cover the following information:

- Understanding the procedures
- Setting up the form page
- Creating rectangles
- Creating a combo box
- Creating text input fields
- Creating a static text field
- Creating check boxes
- Creating command buttons
- Writing simple scripts for a form and a command button

Understanding the Procedures for the Phone Message Form

The phone message form has a graphical design, uses several different types of form objects, and has scripts associated with the form and two layout objects.

The diagram shows a phone message form with various input fields and buttons. Labels on the left side point to specific components: 'Rectangle' points to the top header area; 'Combo box' points to the 'To:' dropdown menu; 'Static text' points to the title 'While You Were Out'; 'Text input' points to the 'Mr./Ms.' field; another 'Text input' points to the 'Phone' field; 'Check boxes' points to the list of actions (Telephoned, Returned your call, Will call again, Please call); and another 'Text input' points to the 'Message' text area. Labels on the right side point to: 'Text input' for the 'Date:' and 'Time:' fields; 'Rectangle' for the area containing the contact information fields; another 'Rectangle' for the area containing the action checkboxes; and 'Command buttons' for the 'Send' and 'Clear' buttons.

Rectangle

Combo box

Static text

Text input

Text input

Check boxes

Text input

To: []

Date: []

Time: []

While You Were Out

Mr./Ms. []

Or []

Phone []

Telephoned ☐

Returned your call ☐

Will call again ☐

Please call ☐

Would like to see you ☐

Was in to see you ☐

Urgent ☐

Sent you a FAX ☐

Message []

Send

Clear

Text input

Rectangle

Rectangle

Command buttons

The procedures in this chapter will show you how to create the layout objects used in the phone message form.

Before beginning the procedures, you need to know that layout objects can be field or non-field objects and that field objects have a field frame, label, and body.

You also need to know how to perform the following actions:

- Move from view to view in the Designer mode
- Move back and forth between the Designer and Filler modes
- Select layout objects
- Open the properties dialog box and move from panel to panel

See Chapters 3 and 4 in this book, and Chapter 4 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information about these topics.

As you're working through the procedures, creating layout objects and dragging them into position, you may find the following tips useful:

- Snap to ruler is active by default. If you want to disable this function, choose View - Rulers and Guides - Snap to Rulers.
- Once you've opened the properties dialog box, drag it to the right side of the Designer window. In this position the properties dialog box can stay open without obscuring your view of the phone message template.
- As you select objects, remember that you can select and deselect the Select Field Frame First check box. To do this, choose File - cc:Mail Forms Preferences.
- Save the form as you work.
- Check the progress of your work by switching back and forth between the Designer and Filler modes. You can click Filler and Designer icons or press F7.
- Work with the sample template MESSAGE.LTM open in another window. This is particularly useful if you want to view the final appearance of an object, or check an object's placement coordinates and use them in your form.

MESSAGE.LTM is in the same directory as the sample form applications. The default path for this directory is C:\DESIGNER\WORK\SAMPLES.

- The instructions in this chapter assume that the unit of measurement is inches. To use inches as the unit of measurement, choose Form - Properties and select English as the measurement system.

Setting Up the Page

The first procedure for setting up a form is to enter a page size, and for this form, a page color.

To set up the page

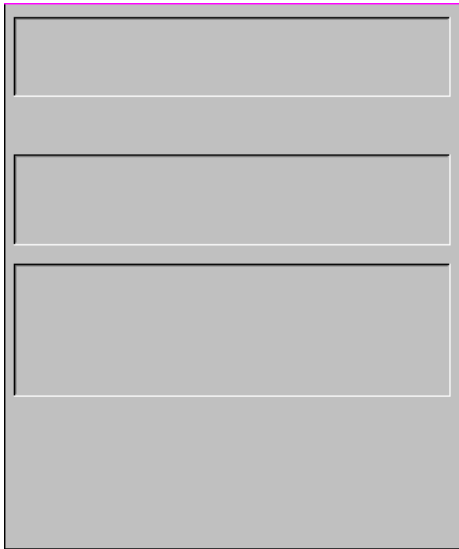
1. In the Form Layout view of Designer mode, open the properties dialog box for the page. (Either double-click on the blank page or choose Page - Properties.)
2. On the Basics panel, under Page Dimensions, type **5.25 x 6.0 inches**.
3. Under Page Color, click to display the color palette.

4. Select a page color for the message form. (The color used in MESSAGE.LTM is Pastel Pink.)

When you're in the palette, hold down the mouse button to display the color names as you move the mouse pointer across the colors. The color name is displayed across the top of the palette. When you release the mouse button, that color is selected and the palette is closed.

Creating the Backdrop Rectangles

The message form uses rectangles to create three-dimensional effects. Other layout objects are then placed on top of the rectangles.



Start by creating and formatting the first rectangle. Then duplicate it to create the other two rectangles. Duplicating a layout object copies the selected object's size and all its properties (except the object name). All you need to do is move and size the duplicated rectangles; cc:Mail Forms Designer gives them unique names.



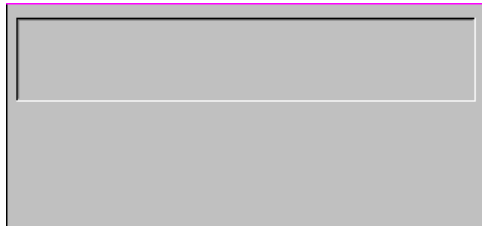
To create the first rectangle

1. Choose Create - Rectangle.
2. Drag to create the rectangle across the top of the page.
3. Double-click the rectangle to display its properties dialog box.
4. Type the following coordinates in the Position panel of the properties dialog box:

<i>For</i>	<i>Type</i>
Top	0.13
Left	0.10
Width	4.00
Height	0.72
Right	4.10
Bottom	0.85

5. On the Style panel (in the properties dialog box), under Style, choose 3D In.

The following illustration shows the first rectangle at the top of the phone message form.



To duplicate the first rectangle to create two additional rectangles

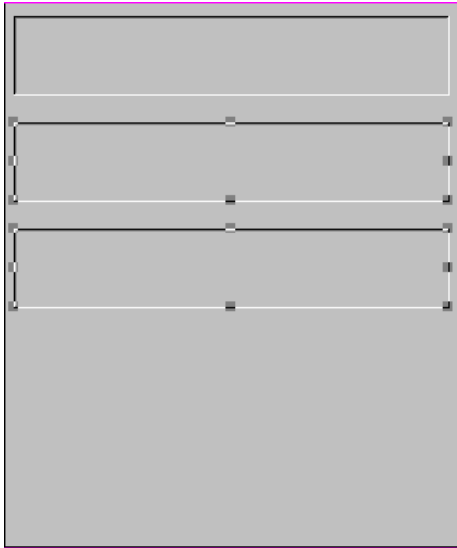
1. Select the rectangle.
2. Choose Edit - Duplicate.

The Duplicate dialog box appears.

3. Enter the following settings.

<i>For</i>	<i>Type</i>
Copies in Horizontal Direction	0
Copies in Vertical Direction	2
Vertical Spacing	0.25

4. Click OK to duplicate the rectangle.



5. Click outside the message form to deselect the rectangles.
6. Select one of the duplicate rectangles, drag it into position, and size it.
Tip Remember that you can open MESSAGE.LTM in another window, and use it as a reference for any object properties, including position coordinates.

7. Repeat step 6 for the other duplicate rectangle.

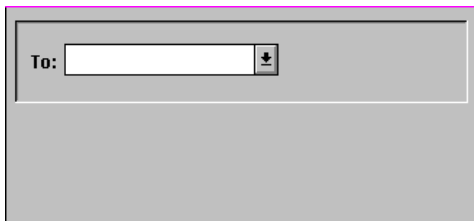
The following illustration shows the three rectangles in place.

A diagram showing a vertical rectangular container with a magenta border. Inside, three identical gray rectangles are stacked vertically, separated by thin white lines. The bottom rectangle is larger than the two above it. This represents the layout of three message form rectangles.

Adding a Combo Box

The combo box will list the names of people to whom a message form can be sent. The person who completes the form selects a name from the list, and Forms Filler sends the message to that person.

In these next procedures, you'll create the combo box and position it on top of the first rectangle.

A diagram showing a gray rectangular form with a magenta border. In the top-left corner, the text 'To:' is followed by a white rectangular input field. To the right of the input field is a small square button with a downward-pointing arrow, indicating a dropdown menu. The rest of the form area is empty gray space.

You'll next place three names on the list. You can also add names using a script, as shown in the scripting procedures at the end of this chapter.



To add the To field combo box

1. Choose Create - Field - Combo Box.
2. Drag to draw and position the field.
3. If necessary, select the combo box's field frame.
4. On the Basics panel of the properties dialog box, type the following:

<i>For</i>	<i>Type</i>
Name	cmbTo
Field Label	To:

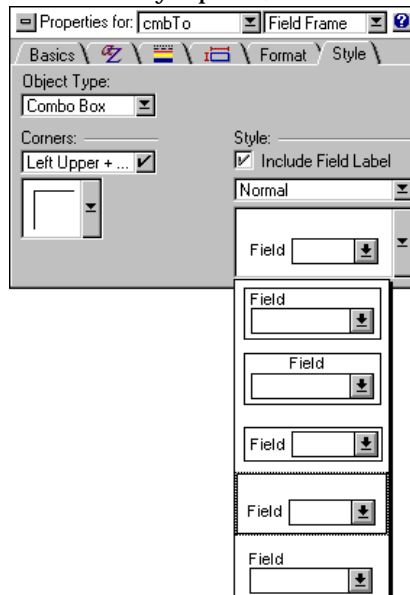
5. On the Font panel, select the following:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

6. On the Format panel, under Type, select String.

Tip Make sure that only Enabled and Visible are checked and that Min Characters and Max Characters are blank.

7. On the Style panel, under the display style, select the fourth option.



Click here to display list

Select this style

8. Select the field body.
9. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

10. On the Pick List panel, select the following properties:

<i>For</i>	<i>Select/type</i>
List Type	List
List	Pam Ressler Liz Murray Matt Mai

The following illustration shows the combo field in place.

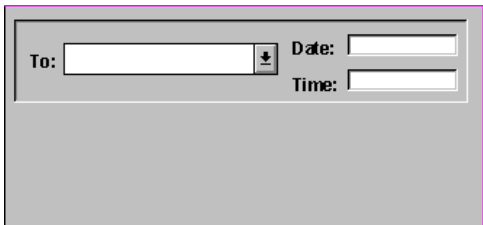
The illustration shows a form layout. At the top, there is a 'To:' label followed by a text input field and a small dropdown arrow button. Below the input field, there are two large rectangular boxes, one above the other, which represent the list box and the filler area respectively. The entire form is enclosed in a light gray border.

11. Choose View - Filler Mode to check your work and see the names in the list box.

Note The names are only displayed in Filler when you (or your users) use the combo box list.

Creating Text Input Fields for the Date and Time

The Date and Time fields are text input fields. These fields are also placed in the first rectangle, and positioned to the right of the To field.

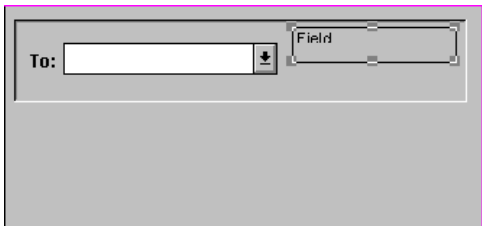


In Forms Filler, when the user first opens this form, the date and time are automatically filled in. This functionality is achieved through scripting, as covered in the last section of this chapter.

In the procedures that follow, you'll first create and format the Date field and then copy and modify it for the Time field.

To create the Date field

1. Choose Create - Field - Text Input.
2. Drag to draw and position the field.



3. Select the field frame.
4. On the Basics panel, type the following properties:

<i>For</i>	<i>Type</i>
Name	txtDate
Label	Date:

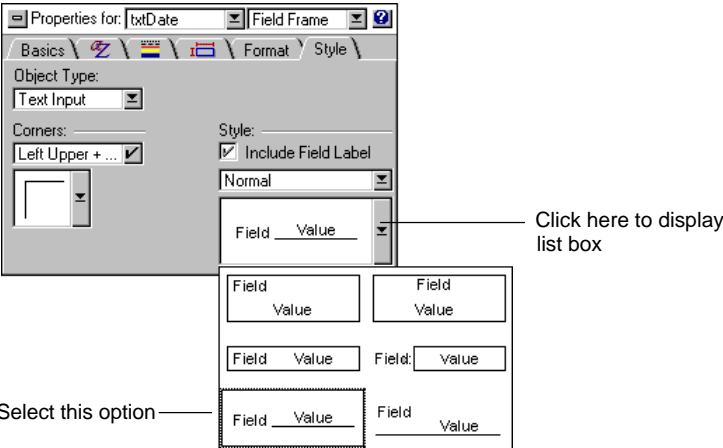
5. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

6. On the Background & Line Panel, under Line Color, select T (transparent).
7. On the Format panel, select the following properties:

<i>For</i>	<i>Select</i>
Type	DateTime
Format	mm-dd-yy

8. On the Style panel, under the display style, select the third option.



9. Select the field body.
10. On the Basics panel, under Text Options, select Single line only (make this the only selected option).
11. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

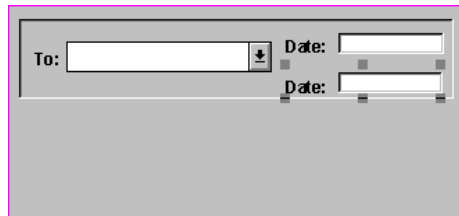
12. On the Background & Line Panel, do the following:

<i>For</i>	<i>Select</i>
Background Color	White
Line Color	T

13. On the Style panel, under Style, select 3D In.

To create the Time field

1. Select the frame of the Date field.
2. Choose Edit - Copy.
3. Choose Edit - Paste.
4. Click below the Date field to paste the copy.

A screenshot of a form with a light gray background. It contains a 'To:' label followed by a text input field and a small downward arrow icon. To the right of this are two 'Date:' labels, each followed by a date picker control. The entire form is enclosed in a thin purple border.

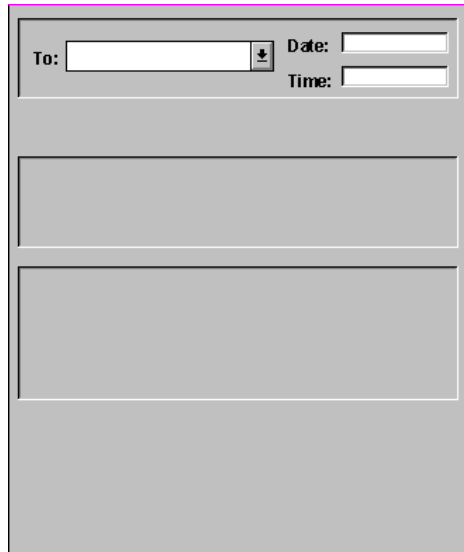
5. Adjust the placement of the field as needed.
6. Select the field frame of the pasted field.
7. On the Basics panel, type the following properties:

<i>For</i>	<i>Type</i>
Name	txtTime
Field Label	Time:

8. On the Format panel, under Type, select String.

Note Make sure that only Enabled and Visible are checked, and that Min Characters and Max Characters are blank.

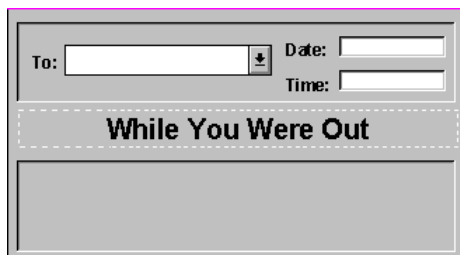
The following illustration shows the Date and Time fields in place.

A screenshot of a form design interface. At the top, there is a 'To:' label followed by a text input field and a small dropdown arrow icon. To the right of this are two stacked input fields labeled 'Date:' and 'Time:'. Below these fields are two large, empty rectangular text areas, one above the other. The entire form is enclosed in a light gray border.

Note The format for the Time field is set in the form's script, as described in the last section in this chapter.

Creating the Static Text Field

The title of this form is “While You Were Out.” Since this is decorative text, it's created as a static text field. This text is placed between the first and second rectangles in the phone message form.

A screenshot of the same form design interface, but now with a static text field added. The text field is located between the two large rectangular text areas and contains the text “While You Were Out” in a bold, black font. The text field has a dashed border. The rest of the form, including the 'To:', 'Date:', and 'Time:' fields, remains the same.



To create the form title

1. Choose Create - Static Text.
2. Drag to fit the text field between the first and second backdrop rectangles, and to run the same width as the two rectangles.
3. Type the text: **While You Were Out**
4. Click outside the text field to end entering text.
5. Select the static text field.
6. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	18
Style	Bold
Alignment	Centered

The following illustration shows the static text field in place.

Creating Text Input Fields for the Mr./Ms., Of, and Phone Fields

The Mr./Ms., Of, and Phone fields are text input fields placed on top of the second rectangle.

The screenshot shows a form titled "While You Were Out" with a dashed border. At the top, there are two rows of fields: "To:" followed by a text input field and a dropdown arrow, and "Date:" followed by a text input field. Below these is a "Time:" text input field. The main title "While You Were Out" is centered in a bold font. Below the title, there are three rows of fields: "Mr./Ms." followed by a text input field, "Of" followed by a text input field, and "Phone" followed by a text input field.

In the first procedure, you create the Mr./Ms. field by copying, pasting, and then modifying the Time field. In the second procedure, you create two duplicates of the Mr./Ms. field, modifying them as needed. In the third procedure, you right-align the field labels.

To create the Mr./Ms. field

1. Select the frame of the Time field.
2. Choose Edit - Copy.
3. Choose Edit - Paste.
4. Click in the upper-left corner of the second rectangle to paste the copy.



The screenshot shows the same form as before, but the "Time:" text input field is now wider, spanning the width of the rectangle below the title. The field is highlighted with a selection box, and the label "Time:" is visible to its left.

5. With the field frame selected, drag to size the field across the width of the rectangle.
6. On the Basics panel, type the following properties:

<i>For</i>	<i>Type</i>
Name	txtName
Field Label	Mr./Ms.

Note Because the label for this field is longer than the label for the Date field, some of the text in the Mr./Ms. label may be obscured by the field body.

7. If necessary, select the field body and drag it to the right to display the entire label.

The following illustration shows the Mr./Ms. field in place.

The screenshot shows a form titled "While You Were Out". At the top, there are two rows of input fields. The first row has a "To:" label followed by a text box and a dropdown arrow, and a "Date:" label followed by a text box. The second row has a "Time:" label followed by a text box. Below these is a dashed-line separator. Under the separator is the title "While You Were Out" in bold. Below the title is a label "Mr./Ms." followed by a text box.

To duplicate the field

1. Select the frame of the Mr./Ms. field.
2. Choose Edit - Duplicate.
3. In the Duplicate dialog box, type the following settings:

<i>For</i>	<i>Type</i>
Copies in Horizontal Direction	0
Copies in Vertical Direction	2
Vertical Spacing	0.04

4. Click OK to duplicate the field.
5. On the Basics panel, change the Name and Label for each duplicate as follows:
 - For the Of field, under Name, type **txtOf**, and under Field Label, type **Of**.
 - For the Phone field, under Name, enter txtPhone, and under Field Label, type **Phone**.

The following illustration shows the three fields in place.

The screenshot shows the same form as before, but with three fields below the title "While You Were Out". The first field is labeled "Mr./Ms." and has a text box. The second field is labeled "Of" and has a text box. The third field is labeled "Phone" and has a text box.

To right align the field labels

1. Select the frames for the three fields (Mr./Ms., Of, and Phone).
2. With the mouse pointer on one of the three fields, click the right mouse button to display the Quick menu.
3. Choose Select Field Label Only.

Now only the labels of three fields are selected.

4. With all three labels selected, choose Edit - Align - Right.

The following illustration shows the finished Mr./Ms., Of, and Phone fields.

A screenshot of a phone message form. At the top, there are three input fields: 'To:' with a dropdown arrow, 'Date:', and 'Time:'. Below these is a dashed-line box containing the text 'While You Were Out'. Underneath this box are three stacked input fields labeled 'Mr./Ms.', 'Of', and 'Phone'. The bottom half of the form is a large, empty rectangular area.

Creating the Check Boxes

The phone message form has the following eight check box objects.

A screenshot showing eight identical check box objects arranged in a 4x2 grid. Each object consists of a rectangular frame containing the text 'Would like to see you' followed by a small square checkbox.

The procedures show you how to create one check box and then duplicate it. This time you have the option of duplicating horizontally and vertically at the same time.

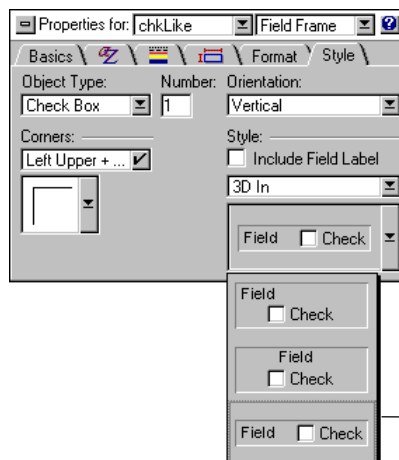
Once the check boxes are duplicated, you'll edit the captions of the duplicates. The editing procedure shows you the **F2 (EDIT)** shortcut. You can also use this shortcut for editing field labels and static text fields.

When you're working with check boxes, keep in mind that the body of a check box has two components: the button and the caption. The button is the box that the user clicks to place or remove a check mark. The caption is the text that describes the setting for the corresponding button. When selecting the field body, you can select either the button or the caption, or both.

The following procedures create a check box with the longest caption first. By fine-tuning the placement of the longest caption with its button, and adjusting the object size, you're assured that the duplicates won't require as much effort.

To create the first check box

1. Choose Create - Field - Check Box.
2. Drag to draw and position the first check box.
3. Select the field frame.
4. On the Basics panel, under Name, type **chkLike**.
Also make sure that Field Label is blank.
5. On the Background & Line Panel, under Background Color, select White.
6. On the Style panel, under Style select 3D In.
7. On the Style panel, under Style, select the last display style option in the list box.



Click here to display list

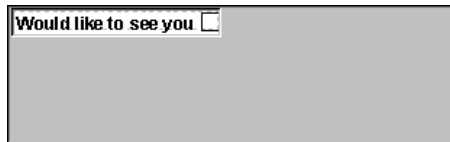
Select this option

8. Select the field body.
9. On the Basics panel, under Caption, type: **Would like to see you**
Tip Type a space before “Would.” The space prevents the text from running into the field frame.
10. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

11. If necessary, drag the field body (button and caption) to the left to fit the caption in the field frame. (You may also need to increase the width of the field frame.)
12. On the Basics panel (for the field body), under Caption Orientation, select Left.

The following illustration shows the first check box.



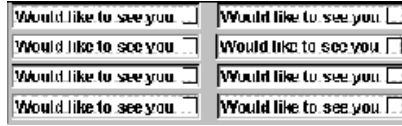
To duplicate the check box

1. Select the frame of the check box field.
2. Choose Edit - Duplicate.
Note The next step creates all the check boxes at once. You may want to first duplicate one check box horizontally, adjust the widths of the two check boxes, and then duplicate the remaining check boxes vertically.
3. Type the following approximate settings (the settings vary depending on the width of your check box):

<i>For</i>	<i>Type</i>
Copies in Horizontal Direction	1
Horizontal Spacing	0.25
Copies in Vertical direction	3
Vertical Spacing	0.05

4. Click OK to duplicate the check box.

The check boxes should look similar to the following:



To edit the names and captions of duplicated check boxes

1. On the Basics panel, change the following field names for the duplicated check boxes. The field names are listed from left to right, top to bottom.

chkTele	chkLike
chkRet	chkWas
chkAgain	chkUrgent
chkPlease	chkFax

Note The following steps show how to use F2 (EDIT) to edit a field body. If you prefer, you can select the field bodies and change the caption in the properties dialog box.

2. Select the caption in the first check box.
3. Press F2 (EDIT).
4. Delete the existing characters and type the new caption, Telephoned (remember to type a space before the caption).
5. Either click in the next caption, or press ↓ to move the pointer to the next caption.
6. Change the following captions. They are listed from top to bottom, left to right.

Telephoned	Would like to see you
Returned your call	Was in to see you
Will call again	Urgent
Please call	Sent you a FAX

7. Click outside a text object to end editing the captions.
The following illustration shows the check boxes in place.

The illustration shows a form for a phone message. At the top, there is a 'To:' field with a dropdown arrow, a 'Date:' field, and a 'Time:' field. Below these is a dashed-line box containing the text 'While You Were Out'. Underneath this box are three stacked text input fields labeled 'Mr./Ms.', 'Or', and 'Phone'. At the bottom, there is a grid of eight checkboxes arranged in two columns and four rows. The first column contains: 'Telephoned', 'Returned your call', 'Will call again', and 'Please call'. The second column contains: 'Would like to see you', 'Was in to see you', 'Urgent', and 'Sent you a FAX'.

Telephoned	<input type="checkbox"/>	Would like to see you	<input type="checkbox"/>
Returned your call	<input type="checkbox"/>	Was in to see you	<input type="checkbox"/>
Will call again	<input type="checkbox"/>	Urgent	<input type="checkbox"/>
Please call	<input type="checkbox"/>	Sent you a FAX	<input type="checkbox"/>

Creating a Scrolling Text Input Field

In Forms Filler, the user types text in the Message field. When this text is longer than the displayed size of the Message field, the Message field displays scroll bars. When the amount of text to be entered in a field will vary, you can specify that the field must have scroll bars. Then, as the developer you don't need to worry about the size of the text field, and the user doesn't have to abbreviate the text to fit a predetermined field size.

The Message field is a text-input field. One of the text option properties that you'll select for this field is to display scroll bars.



To create the Message field

1. Bring the bottom edge of the message form into view in the window.
2. Choose Create - Field - Text Input.
3. Drag to create the field in the lower-left corner of the message form.

4. Select the field frame.
5. On the Basics panel, type the following properties:

<i>For</i>	<i>Type</i>
Name	txtMessage
Field Label	Message

6. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

7. Select the field body.
8. On the Basics panel, under Text Options, select the following three options:
Display Base Lines
Vertical scroll
Wrap text auto

9. On the Font panel, select the following properties:

<i>For</i>	<i>Select</i>
Size	11
Style	Bold

10. On the Background & Line Panel, select the following properties:

<i>For</i>	<i>Select</i>
Background Color	White
Line Color	Black

The following illustration shows the Message field in place.

To: Date:
Time:

While You Were Out

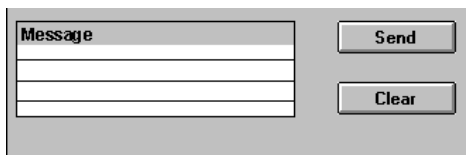
Mr./Ms.
Of
Phone

Telephoned <input type="checkbox"/>	Would like to see you <input type="checkbox"/>
Returned your call <input type="checkbox"/>	Was in to see you <input type="checkbox"/>
Will call again <input type="checkbox"/>	Urgent <input type="checkbox"/>
Please call <input type="checkbox"/>	Sent you a FAX <input type="checkbox"/>

Message

Creating the Command Buttons

This form has two command buttons: the Send button and the Clear button. These buttons are located in the lower-right corner of the phone message form, next to the Message field.

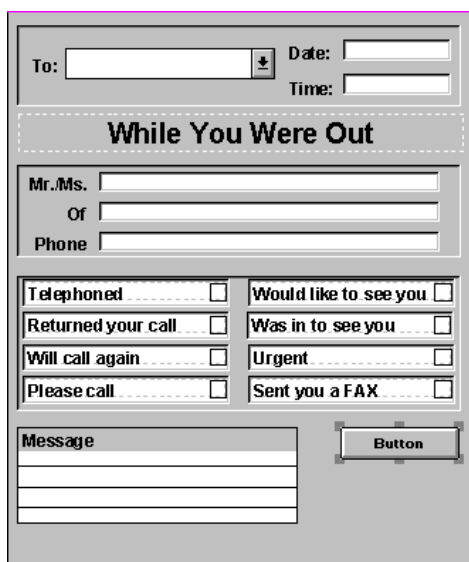


In the following procedures, first you'll create the Send button and then duplicate it to create the Clear button. Each button has a script that specifies what happens when a user clicks the button in Forms Filler. The script for the Clear button is included in the last section of this chapter. (You can view the script for the Send button in the MESSAGE.LTM sample form. See Chapter 14 for more information.)

The font settings for the buttons use the standard Windows typographical convention for buttons, rather than the font used in the other fields.

To create the command buttons

1. Choose Create - Command Button.
2. Drag to draw and place the button in the lower-right corner of the form.



- On the Basics panel, type the following properties:

<i>For</i>	<i>Type</i>
Name	cmdSend
Caption	Send

- On the Font panel, type the following properties:

<i>For</i>	<i>Select</i>
Font	MS Sans Serif
Size	8
Style	Bold

- Duplicate the button.
- On the Basics panel for the new button, type the following properties:

<i>For</i>	<i>Type</i>
Name	cmdClear
Caption	Clear

The following illustration shows the command buttons in place.

Writing the Scripts

The sample form MESSAGE.LTM has scripts for the form and for the Send and Clear buttons.

- The form script adds three names to the To: field's list box, specifies that the Time and Date fields use system dates, and places the initial focus on the To: field. The form script executes when Forms Filler user creates a phone message form.
- The Send button script sends, via e-mail, the phone message form to the person selected in the To: field. This script executes when a user clicks the Send button.
- The Clear button script clears the values from all fields and this script executes when the user clicks the Clear button.

The scripts for the form and the Clear button are included in the following procedures.

See the “Phone Message Form” in Chapter 14 for information on the script for the Send button. Chapters 8, 9, and 10, and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* have details on using the LotusScript language. Also, see Help for information on LotusScript.

To enter the form script

1. In form template, click the Script tab to display the Script window.
2. Under Object, select Form.
3. Under Proc, select NewForm.
4. Type the following script. (This is the same script that appears in MESSAGE.LTM.)

Note When you first display the Script view, Forms Filler displays a default SUB statement. Replace that statement with the SUB statement in the following script.

```
SUB BEFORE1stTo (f1 AS FIELD)

%rem
*****
The first three names in the combo box, cmbTo, were
entered on the Pick List panel.

Additional names can be added using the AddItem method.
Names can be replaced by using the relevant list index.
*****
%endrem

Dim MailList as listbox
```

```

Set MailList = cmbTo.listbox()

MailList.AddItem "Jeff Falco",3
MailList.AddItem "Shane DiCristina",4
MailList.AddItem "Zac Blanchette",5

txtDate.Value=Date

txtTime.Value=Format$(Time,"h:mm AM/PM")

cmbTo.Edit

END SUB

```

To enter the Clear button script

1. In Form Layout view, position the mouse pointer on the Clear button and click the right mouse button.
2. Choose Script.

The Script view appears, ready for you to enter the script for the button.

Note The LotusScript language always identifies an object with the object's Name from the Basics panel in the properties dialog box.

3. Type the following script:

```

SUB BUTTONcmdClear (B1 AS BUTTON)

    Dim f as Form
    Set f=bind form("")
    f.Clear

END SUB

```

See Chapter 14 for a complete discussion of the Phone Message form.

Chapter 6

The Object Library

The object library is a collection of frequently used objects and, when applicable, their associated scripts. Because many of these objects are ready-made, all you need to do is copy them to a form template. The object library includes objects and grouped objects, such as the following examples:

- Send, Next Page, and Clear Form buttons
- Signature fields with time and date stamps
- Signature fields that act as an authorized list of signatures
- An object that amortizes a loan
- A grouped object that allows for the addition of an ODBC data source at runtime

To facilitate looking up and finding objects, the object library stores the objects in categories. For example, two of the categories are Financial Data and Signature Fields.

As an introduction to the object library, the following sections show you how to perform the following tasks:

- Browse the object library
- Copy a library object to a template
- Copy a template object to the library
- Change a library object's category

In addition to these procedures, you can rename and delete library objects, and print an index of all objects or just the objects in a library category.

See Chapter 4 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and search on "object library" in Help.



Browsing Through the Object Library

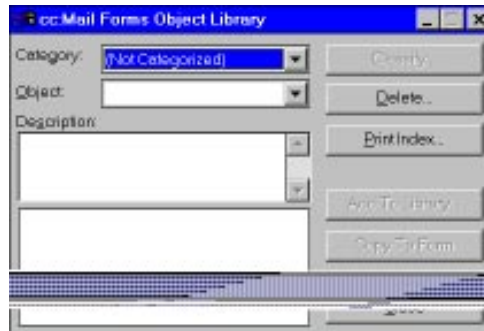
The best way to become familiar with the objects in the object library is to browse through the categories and the objects in each category.

A category is a classification of objects. For example, the Financial Data category includes an object that amortizes a loan and an object that calculates sales tax.



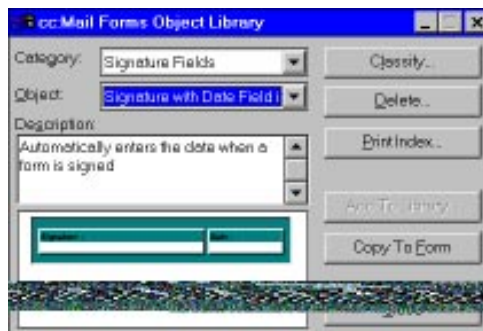
1. Choose File - Object Library.

The Object Library window appears.



2. Under Category, select the category to browse.
3. Under Object, select the object to view.

When you select an object, the object's description and as much of the object as fits in the preview box appears. For example, in the following illustration, the selected object is the Signature with Date object from the Signature Fields category.



4. Click Close to close the Object Library window.

The Object Library window stays open until you close it; you can browse and retrieve as many objects as you want without needing to reopen the window.

You can also collapse the Object Library window by double-clicking its title bar. Double-click the title bar again to expand the window to its full size.

Tip You can print an index of the entire library or just a category. For more information, search on “object library” in Help, or click the Help button in the Object Library window.

Copying an Object from the Object library

You copy an object from the Object library window to the form template in the Designer window. Keep in mind that most library objects have an associated script or scripts. These scripts were written to accommodate the most frequent use of the object. You may need to edit the scripts to suit the needs of your template.



1. Choose File - Object Library to display the Object Library window.
2. Under Category, select the category from which you want an object.
3. Under Object, select the object you want.
4. Perform one of the following actions:

- Click Copy to Form
- Drag the object from the preview box to the template

Note If another object in the template has the same name as the library object, cc:Mail Forms Designer changes the name of the object copied from the library. You may need to change the object name in any scripts associated with the object.

See Chapters 8 through 10 and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information on the LotusScript language.

Copying an Object to the Library

If you've created an object, or a grouped object, that you'll be using again, you can copy it to the library. When you copy an object, cc:Mail Forms Designer also copies the associated scripts.



1. Select the object's frame, or, for a grouped object, select the group.
2. Choose File - Object Library to display the Object Library window.
3. Click Add to Library.

The Classify Object dialog box appears.

4. (Optional) Under Object, type a different identifying title for the object. This title will be displayed in the Object list box in the Object Library window.
5. For the new object's category, perform one of the following actions:
 - Under Category, select an existing category for the object.
 - Under New Category, type the name of a new category into which you want to file the object.
6. (Optional) Under Description, type the description that you want to appear in the Object Library window.
7. Click OK to add the object to the library.

Changing an Object's Category

An object resides in only one category in the library; however, you can change an object's category. For example, you may want to create your own categories, or place frequently used objects in a special category.



1. Choose File - Object Library to display the Object Library window.
2. Under Category, select the object's current Category.
3. Under Object, select the object whose category you're changing.
4. Click Classify.
5. To assign a different category, perform one of the following actions in the Classify Object dialog box:
 - Under Category, select an existing category for the object.
 - Under New Category, type the name of a new category into which you want to file the object.
6. Click OK to change the object's category.

Chapter 7

Routing Lists, Roles, and Stops

A routing list allows a user to automatically route a form through an organization. A routing list is an e-mail path consisting of mail stops, where each stop typically has one recipient. For example, a time sheet form could have a routing list that sends the time sheet from the employee to the employee's manager for approval, and then, once approved, to accounting for payment. The routing list is part of the time sheet's form template.

The scope of a routing list is company-wide if you use roles rather than employee names. A role is a representation of a job function or position, such as Manager, Accountant, Administrator, Vice President, and Telemarketer. Most roles are filled by different people across an organization. For example, the manager who approves Dan Brown's time sheet is not necessarily the same manager who approves Adrian Gray's time sheet. Forms Filler determines who is filling a role by looking up the information in a roles database table. By using roles, you can make one routing list work for all employees across an organization.

You can also specify actions that Forms Filler is to take before mailing (pre-processing actions) a form to the next stop, or before opening (post-processing actions) a form at a stop. For example, Forms Filler can determine who is the next person to receive a form based on conditions that you specify. You can also display or hide fields and enable or disable fields at specific stops. You specify these actions with a pre-processing and/or a post-processing script for a stop.

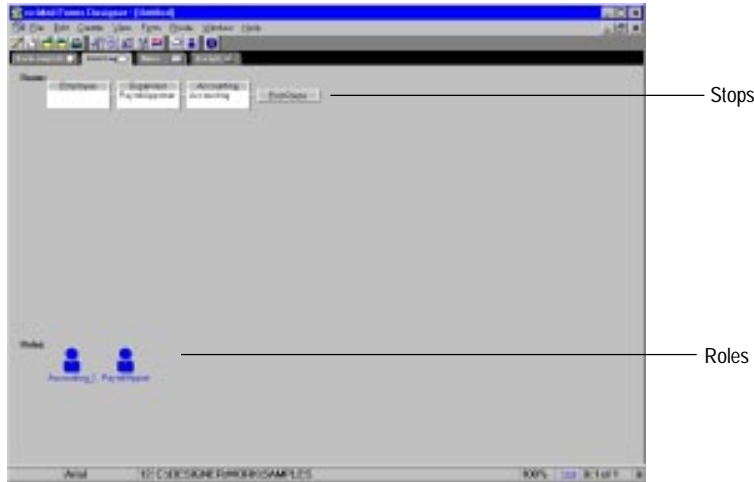
This chapter covers the following topics:

- Routing view
- Roles database table, editor, and script
- Roles in a routing list
- Stops in a routing list

Note Routing in Forms Filler works with cc:Mail™. Forms Filler can access a mail system's address book directly and send and receive forms in that mail system.

The Routing View

The Routing view of Designer mode is where you work on a form's routing list. You use this view to add roles and stops to a routing list. The routing list in the following screen has three stops and two roles.



The roles are Accounting and PayrollApprover, and the stops are Employee, Supervisor, and Accounting. The first step in building a routing list is understanding how Forms Filler uses roles.

To use roles, you set up a database table that defines the role names and who fills the role for each employee. Next, you start the routing list by adding the appropriate roles to the Routing view. Then you add the stops, and for each stop, select the appropriate role as that stop's recipient.

For example, in the preceding screen, the Accounting and PayrollApprover roles are at the bottom of the screen. The first stop is the Employee stop. This is the Originator's stop, and therefore does not have a recipient. The second stop is the Supervisor stop, and the recipient is the person who fills the PayrollApprover role.

When an employee e-mails the form, Forms Filler looks up the role PayrollApprover, and determines who fills this role for the current logged-in user (the employee). Forms Filler then e-mails the form to the employee's payroll approver.

To help you work with roles, Designer supplies a default roles database table, and a default script that performs the database table look-up. See the following section for more information.

Roles Database Table, Editor, and Script

Even if you're not familiar with database tables or scripting, you can use roles in your form. cc:Mail Forms Designer has a default roles database table, and a default script that performs the database look-up. To add your roles to the default database table, you can use the Roles Database Editor.

This section provides information on the following topics:

- The default roles database table
- The Roles Database Editor
- The default script for a role

Viewing the records in the default roles database table

1. Choose File - Open Form Template.
2. Select ROLES.LTM. (The default location for this template is C:\DESIGNER\WORK\SAMPLES.)
3. Click OK.
4. Switch to Filler mode.
5. Click OK to close the message dialog box.
6. To view records, click one of the following arrows or buttons:
 - The left or right arrow to display the previous or next record.
 - Home or End to display the first or last record.

The default roles database table

The default roles database table is ROLES.DB. This table is installed with the cc:Mail Forms Designer sample forms. (The default directory is C:\DESIGNER\WORK\SAMPLES.)

Instead of creating a new roles database table, you can add your own records to ROLES.DB. When you first receive this table, it has the following 19 records that are used by the sample forms.

<i>Fromname</i>	<i>Rolename</i>	<i>Toname</i>
Jeff Falco	Accounting	Pam Ressler
Jeff Falco	POApprover	Liz Murray
Jeff Falco	Scheduling	JeffFalco
Liz Murray	Accounting	Pam Ressler
Liz Murray	Scheduling	Jeff Falco
Matt Mai	Accounting	Pam Ressler

<i>Fromname</i>	<i>Rolename</i>	<i>Toname</i>
Matt Mai	PayrollApprover	Liz Murray
Matt Mai	ReceivingSupervisor	Liz Murray
Matt Mai	Scheduling	Jeff Falco
Pam Ressler	Accounting	Pam Ressler
Pam Ressler	Scheduling	Jeff Falco
Shane DiCristina	Accounting	Pam Ressler
Shane DiCristina	POApprover	Jeff Falco
Shane DiCristina	PayrollApprover	Jeff Falco
Shane DiCristina	ReceivingSupervisor	Jeff Falco
Shane DiCristina	Scheduling	Jeff Falco
Zac Blanchette	Accounting	Pam Ressler
Zac Blanchette	PayrollApprover	Liz Murray
Zac Blanchette	Scheduling	Jeff Falco

The fromname, rolename, and toname fields designate the following addresses or names:

- Fromname is the e-mail address of the current logged-in user.
- Rolename is the role name of the person to whom the form is being mailed, such as PayrollApprover.
- Toname is the e-mail address for the role name. For example, Liz Murray is Matt Mai's payroll approver. If Matt Mai e-mails a form to his PayrollApprover, Forms Filler looks up Matt's payroll approver in the database table and sends the form to Liz Murray.

The Roles Database Editor

The Roles Database Editor is a database front-end application created in cc:Mail Forms Designer. The default values for this application are set to edit the default roles database table. If you are using the ROLES.DB database table that was installed with cc:Mail Forms Designer, you can use the Roles Database Editor to enter and modify your records for the fromname, rolename, and toname fields.



See Chapter 14 for information on how the template was put together, and search on "roles database" in Help.

You use this editor in Filler mode, or in Forms Filler. First create a form using the ROLES.LTM template. Then use the form to modify the database table.

The rest of the chapter describes how to perform the following tasks:

- Create a form with the Roles Database Editor template
- Browse the records in the roles database table
- Add a role (record) to the roles database table
- Update a role in the roles database table
- Delete a role from the roles database table

Tip You can also customize the template for the Roles Database Editor to create your own database front-end application.

To create a Roles Database Editor form

1. Perform one of the following actions:
 - Switch to Filler mode and choose File - New Form.
 - Start Forms Filler. (If Forms Filler is already running, choose File - New Form.)
2. Select ROLES.LTM as the template for the new file. (The default directory for this template is C:\DESIGNER\WORK\SAMPLES.)
3. Click OK to close the message dialog box.

Note A message dialog box appears to explain that this application will modify the default roles database table. If you want to modify a different database table, you need to modify the ROLES.LTM template.

4. Click OK to close the message box.

The Roles Database Editor displays the first record in the ROLES.DB table.

The screenshot shows a form titled "cc:Mail Forms Roles Database Editor" with a teal background. At the top right, it says "Record 1 of 19". Below the title, there are three input fields: "Role:" with the value "Accounting", "From Name:" with the value "Pam Ressler", and "To Name:" with the value "Pam Ressler". To the right of these fields is a control panel with buttons: "<" and ">" for navigation, "Home" and "End" for jumping to the first or last record, and "Clear Fields" to reset the inputs. At the bottom, there are four buttons arranged in two columns: "ADD Current Entry as a New Role", "UPDATE Current Role", "DELETE Current Role", and "DELETE ALL Roles from the Database".

To browse through the records

1. Click one of the following arrows or buttons:
 - The left or right arrow to display the previous or next record, respectively.
 - Home or End to display the first or last record, respectively.

To add a role

1. Click Clear Fields.
2. Type the Role, FromName, and ToName.

Note Remember to observe the database conventions for naming fields. For example, many databases do not allow spaces in field names and accept a maximum of 25 characters for a field name. See the documentation for your database software for more information.

3. Click ADD Current Entry as a New Role.

The role's record is added to the database table.

To update a role

1. Display the record.
2. Edit the Role, FromName, and ToName as needed.
3. Click UPDATE Current Role.

The updates to the role's record are written to the database table.

To delete a role

Caution Keep in mind that the original 19 records in the ROLES.DB table are used by the sample forms. If you delete one of these records, some of the sample forms may not work as expected.

1. Display the record.
2. Click **DELETE** Current Role.

You are asked to confirm the deletion.

3. Click OK to delete the record.

The role's record is removed from the database table.

The default script for a role

When you add a role to a routing list, Designer automatically adds the following script, changing the role's field name to the name you entered in the Routing view. This script looks up the current user's name (fromname) and the rolename in the roles database table in order to determine the toname.

```
SUB ROLERole1 (BYVAL UserName AS STRING, BYVAL RoleName AS
String)
Dim f as FORM
print "The current logged in user is " + username
SET f = BIND FORM("")
dim rolesdb as db
set rolesdb = new db("RolesDB")
dim rs as dbrecords
sqlexp$ = "select toname from roles where fromname = \" +
username +
    \"' and rolename = \" + rolename + \"'"
set rs = rolesdb.executesql(sqlexp$)
if rs.count > 0 then
dim r as dbrecord
set r = rs.getrecord(0)
dim n as string
n = r.getattribute(0)
print "The address found for the role " + rolename + " is " +
n
    f.RoleAddress = n
else
print "Unable to find Role to Route form."
end if
END SUB
```

If you've added your roles in the default roles database table, you do not need to modify this script. Otherwise, make sure to edit the name of the roles database table in the script to the name of the table you're using.

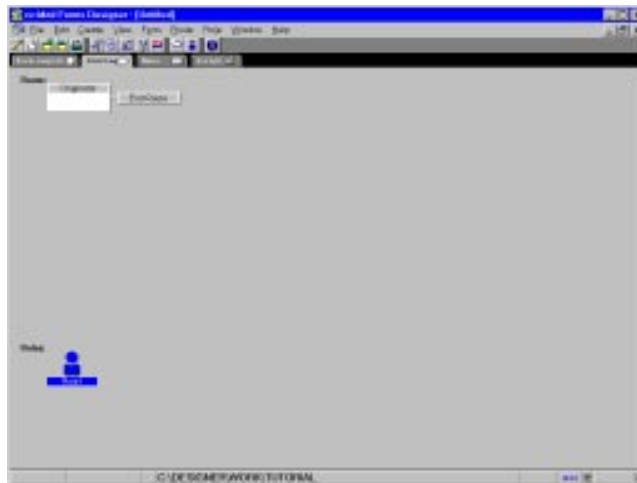
Roles in a Routing List

You specify which roles are in a routing list in Routing view of the template for which you're creating a routing list. You can add roles to a routing list at any time, but they are typically added after the roles are defined in the roles database table.

To add a role

1. Click the Routing tab or choose View - Routing to display the Routing view.
2. Choose Create - New Role.

The new role icon is added to the Routing view, and the default script is added for a role. (The script is visible in the Script view.)

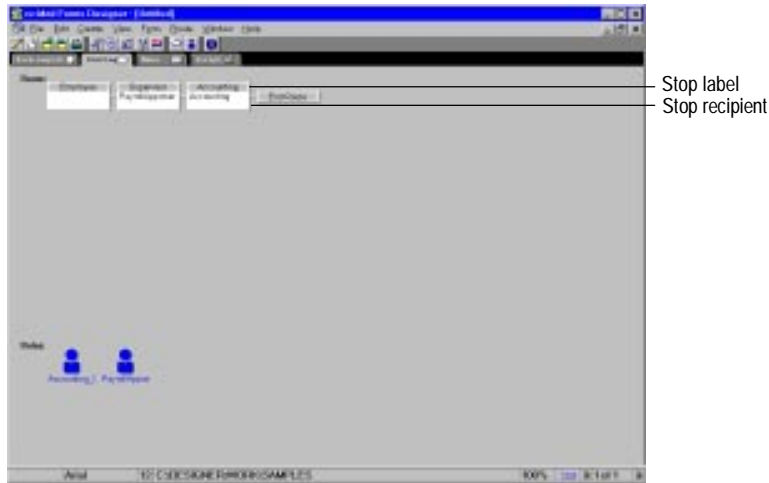


3. Open the properties dialog box for the role.
You display the properties dialog box in Routing view using the same techniques as in Form Layout view; that is, you can double-click the role, choose Role - Properties from the menu, or right-click to display the Quick menu and choose Properties.
4. Under Role Name, type the role's field name as it appears in the roles database table. (For example, if the field name in the table is Accounting, type **Accounting** as the role name in the properties dialog box.)

The role's script is updated with this field name.

Stops in a Routing List

In Routing view, a stop has a label and a recipient. As with other layout objects, you use a stop's label when you write a script for the stop. The recipient is the person who receives the form at that stop. The recipient can be one of the roles displayed at the bottom of Routing view, or a specific e-mail address.



The scripts that you write for stops determine what happens before a form is e-mailed to the next stop or what happens before a form is opened at a stop.

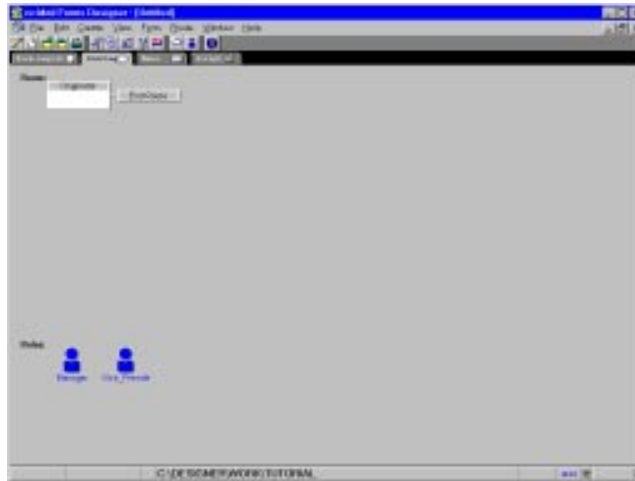
Adding stops to a routing list

Generally, you add stops to a routing list after you've added the roles. When adding a stop, you use the properties dialog box to select the stop recipient and to change the stop label.

A new stop is added after the currently selected stop. If no stops are selected, a new stop is added to the end of the routing list.

To add a stop and a role recipient

1. Click the Routing tab or choose View - Routing to display Routing view.

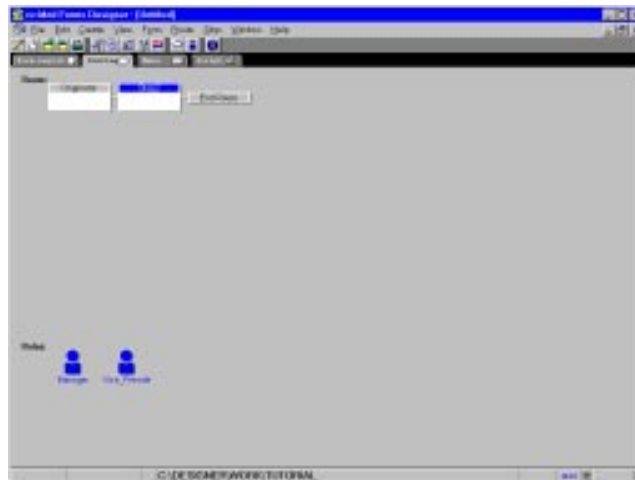


When you first display Routing view, it shows the Originator stop. The Originator is the person who creates the new form in Forms Filler. You add the remaining stops in the routing list.



2. Choose Create - New Stop.

The new stop is added to the routing list.



3. Open the properties dialog box for the stop.

You display the properties dialog box for a stop in the Routing view using the same techniques as in the Form Layout view; that is, you can double-click the stop, choose Stop - Properties from the menu, or right-click to display the Quick menu and choose Properties.

4. If necessary, click the Basics tab.
5. Under Add, select Role.
6. Select the role from the list.
7. Click Add to Recipient List.

The selected recipient appears in both the Recipients list box and in the stop.

8. (Optional) Under Stop Label, type the label that you want to use to identify the stop.



See Chapter 6 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on the other properties you can set for a stop and on selecting an e-mail address for a stop, and search on “stops” in Help.

Pre-processing and post-processing scripts for stops

Forms Filler can execute a script before sending a form to the next stop, and before opening a form at a stop. For example, Forms Filler can execute the following scripts:

- A pre-processing script before opening a form at a stop.
- A post-processing script after the user issues the command to send the form to the next stop, but before Forms Filler e-mails the form.

Pre-processing scripts are used to display and hide fields, and to enable and disable fields. For example, in the Purchase Order sample form application, an approver's signature field is enabled at only the approver's stop. In the New Employee Form sample form, the developer used a pre-processing script to display fields at only specific stops.

Post-processing scripts are particularly useful for conditional routing. For example, in the Purchase Order sample form application, purchase orders of \$10,000 or more must be approved by a group manager and a vice president, and purchase orders of less than \$10,000 can be approved by just a group manager. For the first approver's stop, the developer wrote a post-processing script that, based on the total amount of the purchase order, determines whether the form goes to a second approver or directly to the vendor.



See Chapter 14 in this book for more information on these scripts and examples of other pre-processing and post-processing scripts. For more information on LotusScript, see Chapters 8, 9, and 10 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, and search on "LotusScript" in Help.

Chapter 8

Linking Layout Objects to Database Fields

You can set up your form to read and write information to database tables. For example, you can link a combo box field to a database table, and use a field in that table to populate the combo box's list box. Or you can create a form that serves as a front end to a database table, and that automatically displays the table fields when you open or create a form. You can link as many form fields as needed to as many database tables as needed.

To give you an overview of using databases with cc:Mail Forms Designer, this chapter has the following sections:

- Creating database links
- Working with database link windows
- Linking a pick list to a database table
- Determining when Forms Filler queries a database
- Finding more information on databases

See Appendix D in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on setting up ODBC drivers. cc:Mail Forms Designer is compatible with most databases supporting the ODBC standard.

Creating Database Links

You create a link from a form field to a database table field. To do this, you can use a Database Link window, the properties dialog box, or a script.

Scripting provides the most versatility for creating links to databases. You can write a script to link any field in a form to a database field. The script can include how and when the database is queried, and the information that is read and written to and from the database table. You can use scripting in addition to database link windows and settings in the properties dialog box.

The following sections have more information on database link windows and pick lists.

See Chapters 8 through 10 and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on scripting.

Working with Database Link Windows

A database link window is a straightforward, visual way to link a form field to a database table. The link window appears on top of the Designer window and displays the names of the fields in the database table. When you create a link, cc:Mail Forms Designer draws a link line from the table's field name to the form field.

The following sections provide specific information on working with database link windows:

- Creating a database link window
- Linking a form object to a field name in a database link window
- Hiding database link windows
- Deleting links
- Modifying database records
- Specifying properties for a database link window

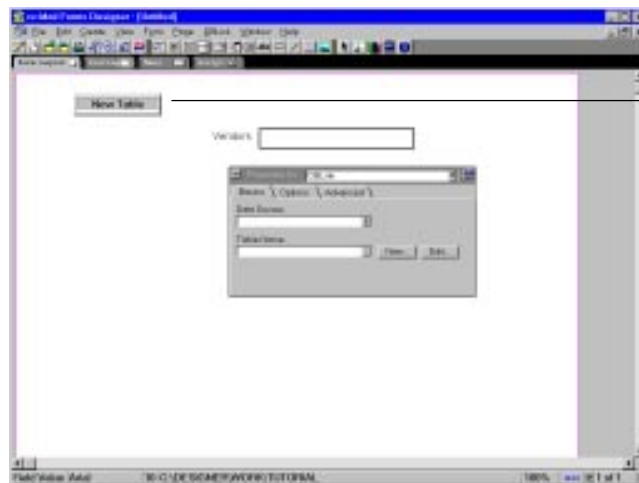
Creating a database link window

A database link window lists the fields in a specific database table. If you plan to create links to more than one table, create a database link window for each table. For example, assume you have one table that lists employee names and another that lists employee titles. To link form fields to both tables, you would create two database link windows: one for the employee names table and one for the employee titles table.



1. Choose Create - Database Link.

An empty link window named New Table appears.



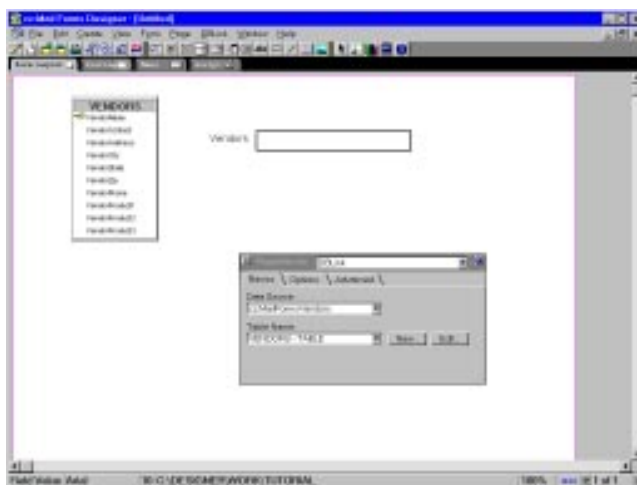
Empty
link window

As with other windows, you can drag the title bar of a database link window to move it to another location. Once you've specified a data source and database table, you can also stretch or shrink the height of the window by dragging one of its borders.

2. Click the link window to select it.
3. Display the properties dialog box.
4. On the Basics panel, select the following properties:

<i>For</i>	<i>Select</i>
Data Source	The data source for the database table
Table Name	The table you want to link to field(s) in the form

The field names from the database table appear in the link window, and the name of the database table appears in the title bar of the link window.



Linking a field layout object to a database field

You can link any field object in a template to a database field. For example, the list for a combo box can display the names of vendors in a Vendors table or the names of employees in a Personnel table.

Note Before creating the link, you must create both the ODBC database source and the database table.

See Appendix D in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on creating an ODBC database source; see your database documentation for information on creating a database table.

To link a field layout object to a database field

1. Create the database link window (as described in the preceding procedure).
2. In the database link window, click the name of the field that you want linked to a form object.
3. Drag the pointer to the form field you want linked.

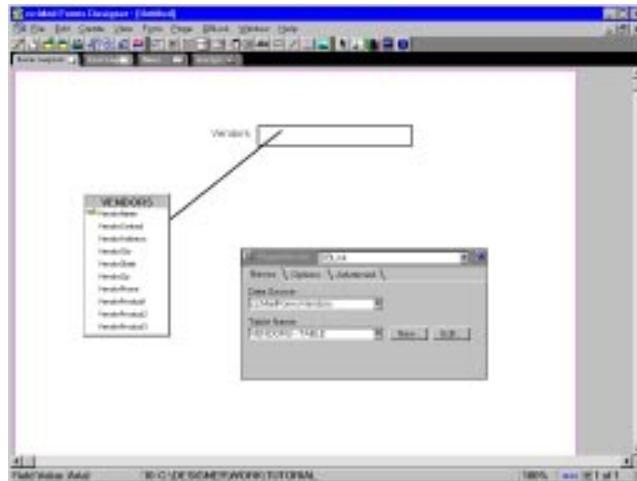
As you drag the pointer, it displays one of two shapes:

The pointer is on an object or area of the form that cannot have a link.

The pointer is on an object that can have a link.

4. Release the mouse button when the pointer is on the field you want linked.

A link line appears between the field name in the link window and the form object.



The layout object's name is changed to the same name as the field in the database table.

Note When you create a field in a form template, the default field label is the same as the field name. If you're using the default field label, the label also changes to reflect the database table's field name.

Hiding a database link window

You can hide the database link windows without losing the links to the database tables.

To hide database link windows, choose View - Show - DBLink Windows.

When the link windows are hidden, the names of the linked fields appear in red in the layout objects (in Designer mode only).

Note Choose View - Show - DBLink Windows to redisplay the database link windows.

Deleting a database link

You can remove a single link to a database table, or remove all links to a database table.

Note When you remove a link, the name of the form field is reset to the name it had when you first created the field.

To remove one link to a database table

1. Select the form field from which you want to remove the database link.
2. Choose Field - Unlink Field.

To remove all links to a database table

1. Click the title bar of the database link window to select it.
2. Choose DBLink - Unlink All.

Modifying database fields from a database link window

When you create a link using a database link window, you can modify, add, and delete fields in the database table from within cc:Mail Forms Designer.

1. Click the title bar of the link window for the table you want to modify.
2. Choose DBLink - Edit Table.

The Table Description dialog box appears.

The Table Description dialog box is shown with the following fields and options:

- Data Source:** LnkFormVendor
- Table Name:** VENDORS
- Buttons:** OK, Cancel, Fields, Add, Change, Edit
- Table Structure:**

Field Name	Type	Length	Nullable
VendorName	Alphanumeric	32	Null
VendorContact	Alphanumeric	32	Null
VendorAddress	Alphanumeric	32	Null
VendorCity	Alphanumeric	32	Null
VendorState	Alphanumeric	2	Null

Below the table, there are input fields for Field Name, Type, Length, and Nullable, each with a dropdown arrow.

3. Select the field to edit.
4. Under Field Name, Type, Length, and Nullable, enter the changes for the field.
5. Click Change to confirm the changes.
6. Click OK to close the dialog box.

Note You can also use the Table Description dialog box to add fields, add and remove keys, and view sample data.

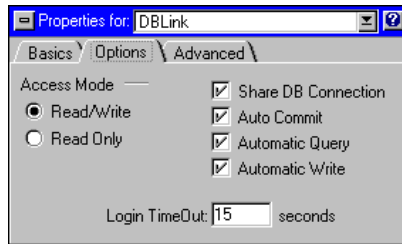
See Chapter 5 in *Lotus cc:Mail Forms Designer Application Developer's Guide*, search on "Table Description dialog box" in Help, or click the Help button in the Table Description dialog box.



Specifying properties for a database link window

The properties dialog box for a database link window has three panels: Basics, Options, and Advanced.

- The settings on the Basics panel specify the data source and database table.
- The settings on the Options panel, shown in the following illustration, determine properties such as read/write access and automatic queries to the database.



- The settings on the Advanced panel determine the transaction isolation level for SQL server transactions.

See Chapter 5 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information on these settings, or click the Help button in the properties dialog box for a database link window.

Linking a Pick List to a Database Table

A pick list is the list of choices displayed in a list box or for a text input field. You can type the list choices in the properties dialog box, or specify that these choices come from a database table. For example, a pick list could list the names from the employee database table or the vendors from the approved vendors database table.

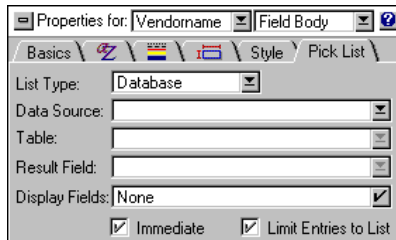
Note Before linking a pick list to a database table, you must create both the ODBC data source and the database table.

See Appendix D in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on creating an ODBC database source; see your database documentation for information on creating a database table.

To link a pick list to a database table

1. Select the body of the field object.
2. Display the properties dialog box.
3. Click the Pick List tab.
4. Under List Type, select Database.

The settings on the Pick List tab change to those you need to create a database link.



5. Select the following properties:

<i>For</i>	<i>Select</i>
Data Source	The name of the data source.
Table	The name of the table to link to the pick list.
Result Field	The name of the table field supplying the field values.
Display Fields	The name of the table field that has the values you want displayed in the pick list.

Note Depending on the needs of your form, the Result Field and Display Fields can have the same field name or different field names. For example, you may want to display the employee names in the pick list, but store the selected employee's ID in the form field. You would select the name of the table field for the employee ID number as the Result Field, and the name of the table field for the employees' names as the Display Fields.

6. Also on the Pick List panel, choose how the pick list is displayed in Forms Filler, and whether cc:Mail Forms Designer accepts an entry not on the pick list.

<i>Option</i>	<i>Select it to</i>
Immediate	Display the pick list when a user positions the mouse pointer on the pick list field. (Deselect Immediate to display the pick list only when a user presses F3 .)
Limit Entries to List	Accept only a value from the pick list. (Deselect this option to accept any value a user enters.)

Note The Immediate and Limit Entries to List options are available only for text input fields and cells in table fields.

Determining When Forms Filler Queries a Database

Forms Filler can query a database when a user creates or opens a form, clicks in a field, enters a value, or specifies the criteria for a search.

You determine how and when a query is executed in a form application. You can set up a query using one of the following methods:

- Specify an automatic query
- Require that a user specifies the criteria for a search
- Write a script

Using automatic query

To specify an automatic query for a database link window, you can select the Automatic Query setting on the Options panel of the properties dialog box. When a user enters data in the form field that is linked to the database table's key field, Forms Filler queries the external database (or databases) for all linked fields. For example, assume the key field is the employee's name. If a user enters an employee's name, Forms Filler automatically queries the database table and fills in all other form fields that are linked to that table.

Running a query from the Forms Filler

A user can also run a query by typing one or more characters in a field that is linked to a database. The user can then choose the Field - Query command, and specify the criteria for a database search.



See Help for more information on running a query from the Filler. Search on “queries,” select the topic titled “Query” then click the Details button in the Query Help topic.

Using a script to query a database table

Scripting provides the most versatility for querying a database table. You can write a script that tells Forms Filler to query a database table in response to any event triggered by a user. For example, you can write scripts that query a database table when the user creates or opens a form, clicks in a field, or clicks a button.

See Chapters 8 through 10 and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on scripting.

Finding More Information on Databases

The extent to which you use databases with form applications depends on your form requirements. For example, you can use databases to perform the following tasks:

- Track a form's progress through a routing list.
You specify the information you want written to the database, and from which stops you want the information written. For more information, see Chapter 9.
- Define the roles you use in a routing list.
The roles database table matches role names with e-mail addresses. See Chapter 7 for information on routing and how to use the roles database table.
- Exchange field information with Lotus Notes.
Information can be written from a form to a Lotus Notes document, and vice versa. For information on how to set up field exchange, see Chapter 11, and search on "field exchange" in Help.

The sample forms included with cc:Mail Forms Designer provide another source of information on databases. Specifically, you may want to review the use of databases in the following form applications:

- In the Application for Employment form, cc:Mail Forms writes information to a database when the user clicks a button. To accomplish this task, the form developer wrote a script for the button.
- In the New Employee form, Forms Filler retrieves existing information from one database table and then writes new information that the user enters to the database table.
- The New Employee and Request for Quotation forms both link the items in a pick list to a database table. In the New Employee form, the developer created the link in the properties dialog box, and in the Request for Quotation form, the developer created the link with a script.
- The Expense Report form demonstrates how to automatically query a database table. When a user enters the employee ID number, Forms Filler queries the database table and fills in the information in all linked fields.

- In the Quotation Record form, a user queries a database by specifying the search criteria.
- Both the Leads Response form and the Roles Database Editor are examples of database front-end applications. When a user opens or creates these forms, Forms Filler displays the first record in a database table. A user can browse through the records, updating them as needed.

See Chapter 14 for detailed information on the sample forms.

Chapter 9

Setting Up a Tracking Database

A tracking database is a database table that collects information from a form as the form progresses through a routing list. The information tracked can range from the date and time a form was delivered to a stop, to information from any field in a form. By tracking information, you always have a form's current status.

To set up tracking for a form, you set up the database table, and then perform two procedures in cc:Mail Forms Designer:

- Specify the information to be tracked
- Enable tracking for the stops in the routing list

The procedures in this chapter use the Routing view to set up a tracking database. An alternative is to write scripts that would perform the same tasks.

Note You can use other ways to track a form's progress through a routing list. For example, you can have a copy of a form sent to an e-mail address outside the routing list, or you can specify that a copy be returned to a specific stop.



See Chapters 8 through 10 and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* for information on the LotusScript language. For more information on tracking a document, search on "tracking a form" in Help.

Specifying the Information to Track

To track information about a form, you must specify the name of the database table where the tracking information is stored and which fields you want tracked.

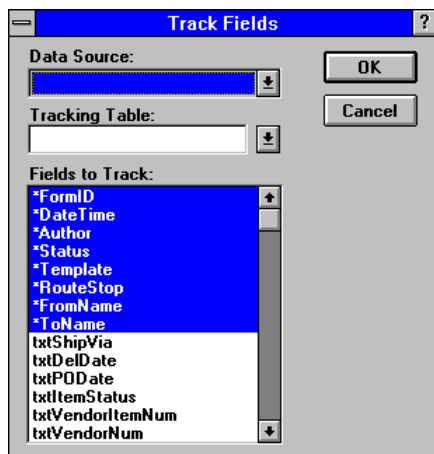
Note Before setting up the database table, you must create the ODBC data source. Although you can create the database table from within cc:Mail Forms Designer, for optimal results create the table using the data source. cc:Mail Forms Designer can approximate the table structure you want, but it's best to set up the table in the data source.

See Appendix D in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information about creating the ODBC data source.

To specify the information to track

1. Display the Routing view in the Designer window.
2. Choose Route - Tracking.

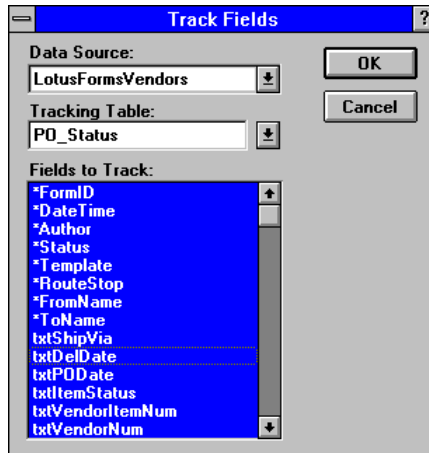
The Track Fields dialog box appears.



3. Under Data Source:, select the data source for the tracking table.
4. Under Tracking Table:, type the name of the database table or select an existing table.

If you select an existing table, new rows are added to the table, leaving the current table information intact, and all form fields that match field names in the table are selected.

For example, in the following dialog box, the tracking table has the same field names as the form, and thus all fields are selected.



5. Under Fields to Track:, select the fields you want tracked. (You can also deselect any of the fields that are automatically selected.)

The Fields to Track: list box lists all fields in the form. The items marked with an asterisk provide information that Forms Filler can record in addition to the fields being tracked. To remove any of these items from the tracking database table, deselect them.

6. Click OK to close the dialog box.

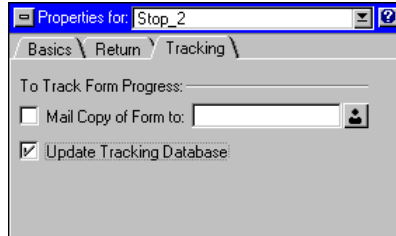
Enabling Tracking for the Stops

Once the tracking database information is set up, you need to specify from which stops Forms Filler is to write information to the database table. You enable tracking on a stop-by-stop basis. When you enable tracking for a stop, make sure that Forms Filler has access to the tracking database from that stop, and that the stop recipient has read and write access to the database table.

Tip If you are using Lotus Notes for the tracking database table, replicate the database, and place replicas at locations that otherwise would not have access to the database table.

To track information from a stop

1. In the Routing view, select the stop from which you want information written to the tracking database.
2. Display the properties dialog box for the stop.
3. On the Tracking panel, select Update Tracking Database.



4. Repeat steps 1 – 3 for each stop you want tracked.

Advanced Uses of Tracking Databases

The uses of a tracking database are not limited just to tracking a form's progress. For example, the developer of the Weekly Time Sheet sample form application specified a Lotus Notes database as both the tracking database and the database that provides the view of the tracking status in Notes. The Notes database also contains a macro that determines when a time sheet is being held at a supervisor's stop and then e-mails a reminder to the supervisor to approve the late time sheet.

You could also set up a tracking database for field exchange with Lotus Notes. For example, you could create a form that is an application for a mortgage. As the application progresses through the approval process, the loan manager can open a Notes database to view the current status of the application. With field exchange, the loan manager could also enter information directly into the application, thus speeding up the approval process.

See Chapter 11 for information on setting up field exchange between Forms Filler and Lotus Notes. Also refer to Chapters 8 and 14 for more ideas and examples of using databases.

Chapter 10

Customizing Menus

A form can have its own customized menu that appears in the Forms Filler. You can add and delete commands, change the order of commands, and enable, disable, and gray commands. You can add a pull-down menu to the main menu, and add a cascade menu to a pull-down menu. The changes to the menu are saved as part of the template.

This chapter describes Menu view and the following ways to customize menus:

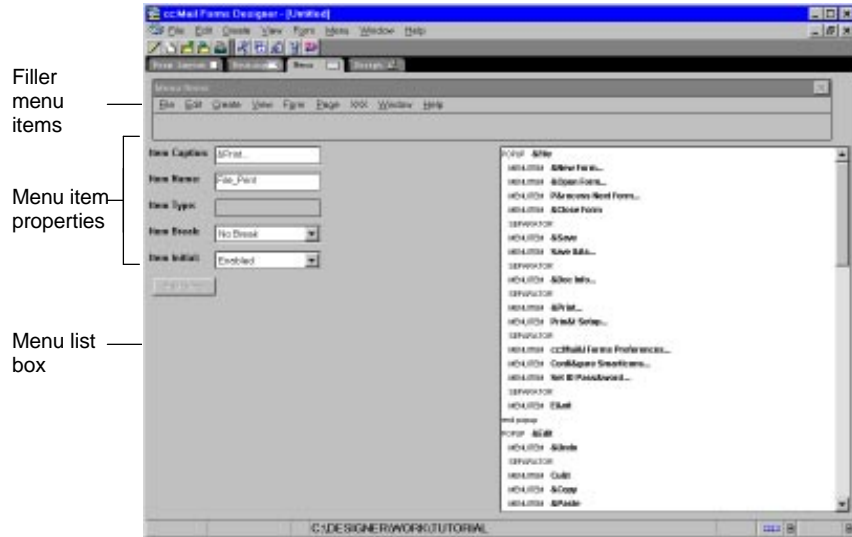
- Add a menu command
- Remove a menu command
- Remove a main menu command
- Reinstate a command

Note A script specifies the actions that Forms Filler executes when a user chooses a customized menu command. (You cannot alter the script for a standard cc:Mail Forms Designer command.)

See Chapters 8 through 10 and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information on scripting. For additional information on customizing menus, refer to Chapter 7 in *Lotus cc:Mail Forms Designer Application Developer's Guide*.

Understanding Menu View

You use the Menu view to create custom menus for a form. Menu view displays the Forms Filler menu items, the properties for each menu item, and the menu list box.



The Forms Filler menu items are a sample view of the Forms Filler menu. As you change the menu items, this sample menu reflects the changes.

The menu item properties are the settings for each menu command and main menu command.

<i>Property</i>	<i>Description</i>
Item Caption	The spelling and capitalization of the menu command as it appears in Forms Filler. The caption can also include an ampersand (&) to designate the command's ALT+letter sequence.
Item Name	The name of the command as it appears in a script.
Item Type	Displays a description of the item type.
Item Break	Designates whether the pull-down or cascade menu breaks into a two-column menu. The two columns can also be separated by a vertical bar.
Item Initial	Specifies whether the initial value of a command is Enabled, Disabled, or Grayed.

The menu list box displays all items in the Forms Filler menu: main menu commands, menu commands, beginnings (pop-up) and endings (end pop-up) of pull-down and cascade menus, and horizontal separator bars between menu commands. You use the list box to specify the location of new commands, and to select items to modify or delete.

Adding a Menu Command

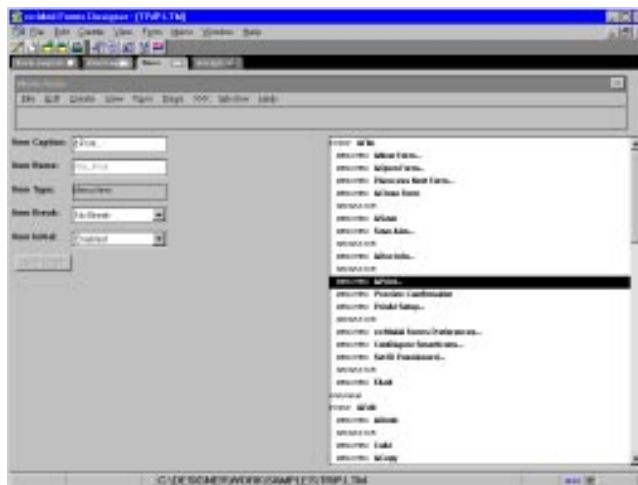
You can add a menu command to a pull-down menu. For example, in the Trip Reservation sample form, the developer added the Preview Confirmation command to the File menu.

The following procedure shows how the developer inserted this new command. The procedure also adds the ALT+V key sequence to the command. This means that a Filler user can press ALT+F+V to select the File - Preview Confirmation command.

See Chapter 14 for information on how the developer used the Preview Confirmation command to open another form.

To add a menu command

1. Click the Menu tab or choose View - Menu Design to display the Menu view.
2. In the menu list box, select **MENUITEM &Print**.



The new command is inserted after the selected command, MENUITEM &Print.

3. Choose Create - Custom Menu Item.

The entry in the Item Caption field is highlighted.

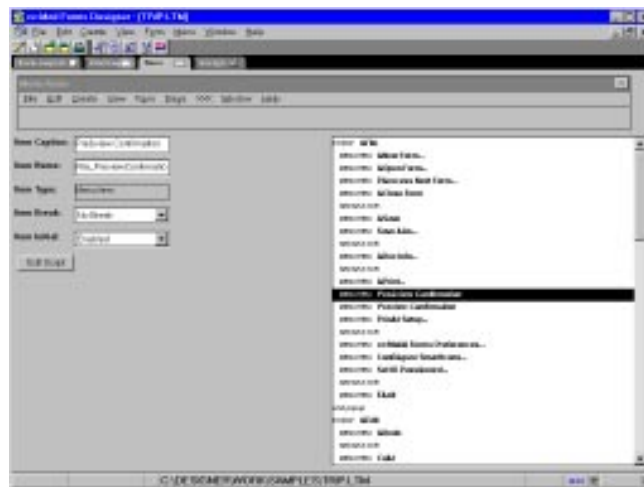
4. Under Item Caption, type **Pre&view Confirmation**.

This is the text of the command as it will appear in the menu (but without the ampersand). The ampersand (&) tells the Forms Filler that the next character is the ALT sequence letter (in this case, v).

The command is added to the menu list box and to the sample menu. If you select File in the sample menu, the Preview Confirmation command is in place.

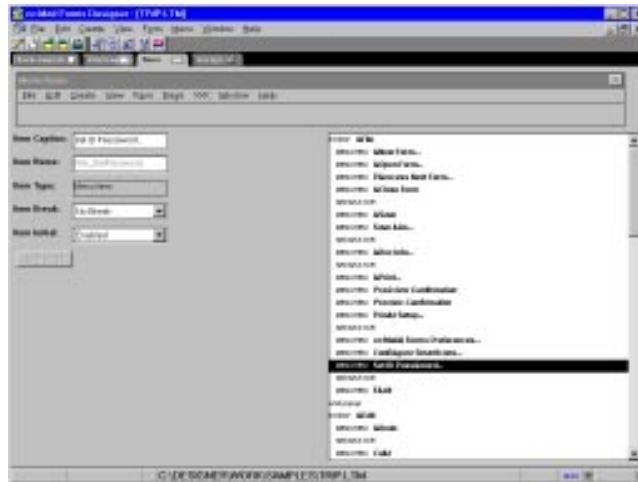
5. Under Item Name field, type **File_PreviewConfirmation**.

This is the menu item name that is used in scripting.



You can remove a command from a pull-down or cascade menu. The following procedure deletes the Set ID Password command from the File menu.

1. Click the Menu tab or choose View - Menu Design to display Menu view.
2. In the menu list box, select MENUITEM Set ID Pass&word.

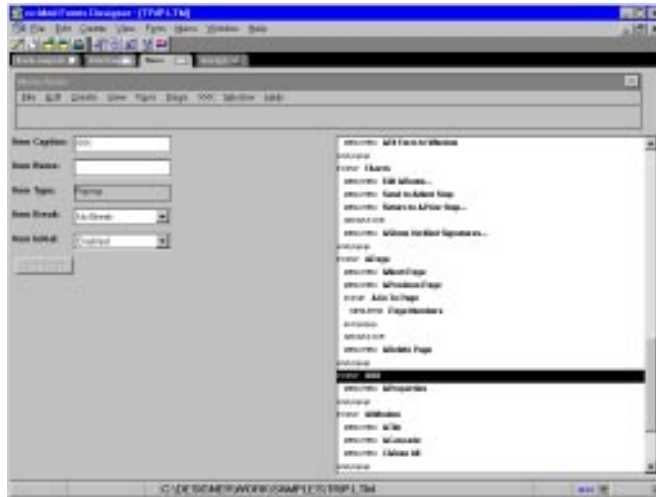


- 3. Choose Edit - Clear.**

Removing a Main Menu Command

A main menu command is a menu item that appears in the menu bar. When a user chooses a main menu command, the command typically displays a pull-down menu. When you delete a main menu command, the pull-down menu is also deleted. The following procedure removes the XXX main menu command.

1. Click the Menu tab or choose View - Menu Design to display the Menu view.
2. In the menu list box, select Pop-up XXX.



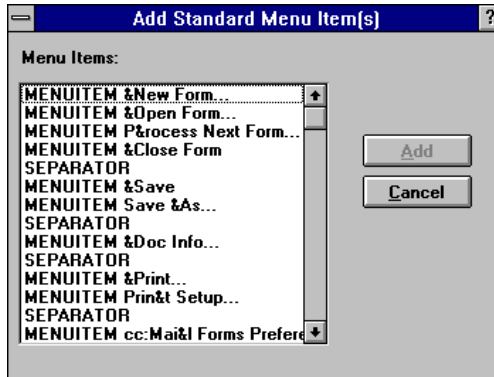
3. Choose Edit - Clear or press DELETE.

Reinstating a Standard Command

If you delete a standard cc:Mail Forms Designer command, you can reinstate the command. This feature is also useful if you decide to move a command; you can delete the command and then reinstate it in another location.

1. Click the Menu tab or choose View - Menu Design to display the Menu view.
2. In the menu list box, select the command or separator that is to appear before the command being reinstated. (The command is inserted after the command or separator you select.)
3. Choose Create - Standard Menu Item.

4. You see the Add Standard Menu Item(s) dialog box.



5. Select the menu item to add.
6. Click Add.

Chapter 11

Using cc:Mail Forms Designer and Notes/FX

The graphical interface of cc:Mail Forms Designer can be combined with the database features of Lotus Notes. You can embed a form in Lotus Notes, and you can exchange data between fields in a Lotus Notes database and fields in a form.

This chapter covers the following information:

- An overview of Notes/Field Exchange (Notes/FX™).
- An example of how to exchange data between cc:Mail Forms Designer and Notes by embedding a form in Lotus Notes and formatting fields for Notes/FX.



Note For an online example of field exchange and embedding a form in Notes, run the Receiving Summary sample form. For information on running this form, see “Sample Forms” in Help.

About Notes/FX

Notes/FX uses Object Linking and Embedding (OLE) to automate data exchange between customized OLE server applications and Lotus Notes data fields.

Information exchanged between Lotus Notes and cc:Mail Forms Designer can be displayed in a Notes database view. This gives you the power of a Notes database to browse and find information collected through a form. In addition, since Notes/FX allows bi-directional exchange, information that you or your users enter in either Forms Filelr or Notes appears in the other application.

For example, you can create a mortgage application form in cc:Mail Forms Designer, specify the fields with the Notes/FX format, and then embed that form in Notes. As the mortgage form goes through each phase of approval, the field data can be written to a Notes database. To check the status of a customer's application, display the appropriate Notes database view. You can then double-click the customer's record to launch the Forms Filler and display the customer's application in its current state. Alternatively, you can type information into the Notes view and then launch Forms Filler, and the new information appears in the customer's application.



See “Share cc:Mail Forms Designer Data Between Applications” in the How Do I? section of Help for more information about Notes/FX.

An Example of Notes/FX

The following example creates a form with fields that have the Notes/FX format, and embeds that form in a Notes form. The information from the form is maintained in a Notes database where it can be viewed without needing to launch Forms Filler. Or, you can double-click a record to launch the Forms Filler and open the form.

You'll follow these steps as you build an example application:

- Create the form
- Create the Notes database, form, and view
- Test the application

To create the form

1. Start cc:Mail Forms Designer.
2. Create five text input fields on the form template.
3. Assign the following properties to the fields:

<i>Name</i>	<i>Field Label</i>	<i>Format</i>	<i>Field Options</i>
LOG_DATE	Date	DateTime mm-dd-yy	Enabled, Visible, Notes/FX
LOG_NAME	Name	General	Enabled, Visible, Notes/FX
LOG_ID	ID Number	Number	Enabled, Visible, Notes/FX
LOG_COST	Cost Center	Number	Enabled, Visible, Notes/FX
LOG_TIMEIN	Time In	DateTime hh:mm:ss AM PM	Enabled, Visible, Notes/FX

4. Choose File - Save As and save the form template in C:\FILLER\WORK (or the equivalent path on your system).
5. Exit cc:Mail Forms Designer.
6. Start Forms Filler in cc:Mail .
The Open Form dialog box appears.
7. Select the form you just created.
8. Choose File - Save As to save the form.
9. Minimize, but do not exit, the Forms Filler.

Note If you exit the Forms Filler, the views in Lotus Notes will not work correctly.

To create the Notes database

1. Start Lotus Notes.
2. Choose File - New Database to create a new database that will exchange the information collected in the form.
3. Make the following selections in the New Database dialog box:
 - Under Template, select Blank.
 - Under Server, select Local.
 - Under Filename, type the path and file name of the Notes NSF file you are creating.
For example, **X:\NOTES\PERSONAL\TIMELOG.NSF** (where X represents the drive where Notes is installed).
 - Under Title, type the title that is to appear on the database in the Notes workspace; for example, **Time Log**.
4. Click New to create the Notes database.

To create the Notes form

1. Choose Design - Forms.
2. Click New.
3. (Optional) Type **Fields from cc:Mail Forms:** and press ENTER.
This is a static text label that serves as informational text only.
4. Choose Design - New Field.
5. Select Create shared field that can be used in other forms.

6. Click OK.
The Share Field Definition dialog box appears.
7. For Name, enter **LOG_DATE** and click OK.
This places a new field on the Notes form.
8. Click to the right of the field and press ENTER.
9. Repeat steps 4 through 8 for the remaining fields: LOG_NAME, LOG_ID, LOG_COST, and LOG_TIMEIN.
10. (Optional) Type **Double-click to edit:** and press ENTER.

To embed the cc:Mail Forms Designer form

1. Choose Edit - Insert Object.
2. Under Object Type, select cc:Mail Forms.
3. Click Choose File.
4. Select the .LFM file and click OK.
5. Click Display Format.
6. Select Bitmap and click OK.
7. Click OK in the Insert Object dialog box.
The Forms Filler appears.

To establish the Notes/FX exchange

1. Minimize cc:Mail Forms Designer and return to Notes.
2. Choose Design - Form Attributes.
3. Under Name, type **REPORT_LOG**.
4. Select only the following options in the dialog box:
 - Include in Compose Menu
 - Default database form
 - Store form in documents
5. Click Object Activation.
6. Select Composing a new Document with this form.
7. Select Hide Notes Document.
8. Click OK twice.
9. Press ESC and click Yes to save the form.

To create the Notes view

A view is necessary to maintain the information.

1. Choose Design - Views.
2. Click New.

Note You must create the same fields created in the cc:Mail Forms Designer form; that is, LOG_DATE, LOG_NAME, LOG_ID, LOG_COST, and LOG_TIMEIN.

3. Double-click the # (pound sign) field.
The Design Column Definition dialog box appears.
4. Under Title, delete the # (pound sign) and type **Date**.
5. Under Formula, delete @DocNumber and type **LOG_DATE**.
6. Under Width, type **10** and click OK.

Notes displays the Date column.

7. Click to the right of the Date Column.
8. Select Design - New Column.

The Design Column Definition dialog box appears.

9. Enter the following:
 - Under Title, type **Name**.
 - Under Formula, type **LOG_NAME**.
 - Under Width, type **15** and click OK.

10. Repeat steps 8 and 9 for the remaining fields in the form, using the following values:

<i>Title</i>	<i>Formula</i>	<i>Width</i>
ID Number	LOG_ID	8
Cost Center	LOG_COST	10
Time In	LOG_TIMEIN	6

11. Press **ESC** and select Yes to save the view.

The Save View dialog box appears.

12. Type **Report Log** and click OK.
13. Press **ESC** to close the database.

To test the application

This procedure tests the exchange of information between Forms Filler and Lotus Notes.

1. Double-click the Notes database to open it.
2. Choose View - Report Log.
3. Choose Compose - ReportLog.

This launches the Forms Filler and opens the form.

4. Fill in the fields with any data.
5. Choose File - Update Lotus Notes.
6. Choose File - Exit & Return to Lotus Notes.

This returns to Lotus Notes and updates the current view with the new information.

Note In the process of creating the Notes form, you selected the Object Activation option “Composing a new Document with this form.” This means that Forms Filler is launched *only* when you create a new form by choosing Compose - ReportLog from the Notes menu. When you double-click the embedded cc:Mail Forms Designer icon, Notes displays only the embedded form.

Chapter 12

Designing Forms

This chapter provides an overview of how to design a form. As you work with these guidelines, also refer to the sample forms to see how they follow these guidelines.

This chapter provides guidelines for the following tasks:

- Developing a form template.
- Creating a database table.
- Distributing a form and its template.

Guidelines for Developing Form Templates

Before you start designing a form, determine what it should do. Identify and interview some of the form's potential users to find out what they need and how they want it presented. It's also a good idea to write a design specification so that you and the client understand what you're developing.

Make sure you have the following information before you start:

- Name and purpose of the form
- Names, functions, and descriptions of all layout objects and menus
- All scanned images (as .BMP and/or .TIF files) to include in the form
- Database sources
- Illustration of the route of the form
- Definition of the next form to be completed (if applicable)
- Descriptions of LotusScript code
- Printing requirements

Starting the form template

Once you've created a design specification, choose a starting point for the form. Define the goal of the form:

- Automation of an existing paper form.

This is an electronic version of a paper-based form. If possible, start out by scanning the existing form. Then use the scanned image as the blueprint for the electronic form, tracing the fields and other form components over the scanned image. When you're finished, delete the scanned image. Once a user completes the form, it can be printed onto its paper-based counterpart.

In most instances the resulting form will be larger than the screen, and the users will scroll the form as they complete it.

- More efficient automation of an existing data-entry process.

This type of form often entails redesigning an existing form (or set of forms) to give it a graphical user interface (GUI). One benefit of a redesign is that each section of a form can be designed to fit the screen. The form can then have buttons and custom menus for navigating from section to section (page to page).

For printing, you can use the LotusScript language to re-create the paper-based form, transfer information from the online form to its paper-based counterpart, and print the form.

- Database front-end that reads and writes information to a database.

This type of form may not require maintaining any copies of the form data, electronic or printed. This form can also be a replica of an existing paper form or a new graphical user interface for entering data.

Designing new templates

As a basis for the design of new form templates, take advantage of any printed forms your company already uses. Keep in mind that time, money, and effort have been invested in creating usable and efficient forms, and that the employees already know how these forms are structured and how to complete them.

When you're working from an existing form, look at it from a layout perspective rather than viewing it as a collection of data input fields. For example, ask yourself the following questions:

- Does the form have sections?
- What is the look of each section?
- Are the fields the same size or type?
- Which sections look enough alike to be duplicated and then edited for each section's specific information?

Overall, remember to keep the design simple, and minimize the information presented to a user at any given time. If necessary, separate the information into a multiple-page form.

The following sections offer additional recommendations.

Uniformity guidelines

Once you've defined the sections of a form, look at the fields in a section to see if they are close enough in design to be copies of each other.

The Split tool, on the SmartIcons bar, is useful for creating a uniform look by splitting existing fields. For example, you can create a text input field and define the field's properties and then use the Split tool to create multiple copies of the field. Each split field inherits the properties of the original field, thus saving you repetitious work in creating each field.

Another option is the Edit - Duplicate command. This command creates horizontal and/or vertical copies of the current layout object.

Color guidelines

In general, the lighter the background color of a form, the better.

- Against a light background color, black text provides the greatest contrast and is the easiest color to read.
- Don't overuse color. You may want to group sets of fields by using the same color for their static text. However, if there are several different-colored text fields on a template, there's no particular emphasis on any one field.

Text typeface, size, and attribute guidelines

Remember that the typeface, size, and attributes of text, as well as its position on the form, all affect the form's usability.

- Choose fonts carefully, and use them consistently. For example, make all field captions the same typeface, size, and attribute.
- The default font size for fields is 10 points. For most typefaces, this is an easy size to read.
- Avoid a cluttered look by using consistent spacing when aligning text and fields, limiting the number of typefaces on a form to one or two, and using a limited variety of text attributes.
- Leave plenty of vertical space between lines of text.
- Keep in mind that not all printers support all fonts. Select fonts that are supported by the users' printers.

Graphics guidelines

You can import a supported graphic file into any template. For example, you can place the company logo on a form by importing a .BMP or .TIF file of the logo. However, keep in mind that it can take substantially more time to display and print a graphic form than a text-only form.

- First scan graphic images to the desired size and then import them.
- Create the graphical user interface elements (fields, list boxes, option buttons, etc.) before creating the navigation elements (buttons, menus).

Security and signature fields

Several ways can be used to secure or hide information in a form.

- With signature fields, you specify which fields are linked to the signature. Whether or not these fields can be edited depends on conditions you set for the signature field.
- A field and its data can be hidden or visible on a form. As a form progresses through a route, you can have certain fields hidden for some stops, and visible for others.
- To protect a form template, you can password-protect it when you save it. This prevents anyone from accidentally modifying it.

Database considerations

If any fields are linked to a database or the form reads or writes information to a database, remember that all users need access to the database. If you are using a Lotus Notes database, you may want to replicate the Notes database to make it accessible throughout a form's route.

- One form can read, write, and be linked to multiple ODBC database tables and sources.
- You can create a script to write information to a database as a form moves along a route.

See “Guidelines for Creating a Database Table” in this chapter for general information on database tables.

Routing guidelines

Once you set up a routing list, you can specify, on a stop-by-stop basis, whether recipients can change the routing list. You can also specify conditional routing where an action or data entered at one stop determines which stop is next.

- For a roles database, define all users and their roles with respect to the application.
- Place the roles database table on a server that cc:Mail Forms Designer has access to as the form moves along the route.
- If you'll be routing the form using Notes, an alternative to a roles database is defining the roles in a Notes name and address book.
- You can write pre-processing and post-processing scripts that determine actions to be taken before and after users complete a form. For example, a pre-processing script could display previously hidden fields, and a post-processing script could determine which stop next receives the form.
- For some forms, allowing the users to modify a routing list provides more flexibility in the route. For example, if the manager who normally approves a form is not available, the user could reroute the form to another manager.

Testing the form

Before releasing the form to your users, test the template and the form to make sure all fields, scripts, and routing tasks perform correctly. By switching between Designer and Filler modes, you can test the form as you build it. You should also test the form from the Forms Filler in cc:Mail, to complete and route the form as your users will.



See “Distributing a Form” in the How Do I section of Help for details on releasing forms to users.

Guidelines for Creating a Database Table

Following are guidelines for constructing database tables. As a general rule, the first two guidelines apply to all tables. The remaining three are applicable only as needed.

1. Create separate tables for related pieces of information and use the same primary key for the tables; that is, don't put all the information in one table.

For example, for a human resources department, one database table might have an employee's personal information such as name, address, and home phone number. Another table would have information such as job level, job code, and salary. Both tables could use the employee's ID number as a primary key.

2. Create a separate table for information that is redundant and repeated in a table.

Using the human resources department again as an example, assume a table lists all employees in a department, and that the table includes each employee's job code and job description. For example, job code 2033 is administrative assistant, and job code 1045 is billing clerk. Instead of having both the job codes and job descriptions for all employees in one table, create a separate table for the job descriptions.

3. Create a separate table for information that does not rely on the primary key; that is, records should contain only information that is dependent on the primary key.

Using a company that markets adventure vacations as an example, assume there is a table for each expedition that lists all available lodging information (hotels, hostels, camp sites, and so on) for that expedition. Because the record information is dependent on the

primary key, if an expedition is discontinued, the corresponding lodging information is no longer available in the database. Instead, place lodging codes in the expedition table, and set up separate tables for the hotels, hostels, and so on.

4. Separate tables that include one-to-many and many-to-many relationships. An adventure-vacations company provides the following examples:
 - An example of a one-to-many relationship is one tour guide who is qualified to lead several different types of expeditions.
 - An example of many-to-many relationships is several tour guides who are qualified to lead the same types of expeditions.

A tour guide's table lists the codes for the expeditions that guide is qualified to lead. Separate expedition tables list the skills required to qualify as a tour guide for each expedition.

5. Separate relationships that are logically related, but that on a practical level are easier to maintain in separate tables. This is useful if some of the data is frequently updated with insertions and deletions.

For example, the table for kayaking expeditions might include information on all the supplies needed for a kayaking trip as well as the suppliers. However, as safety regulations change and as suppliers discontinue certain lines or expand to include new lines, it would be easier to have three tables. One table would have the expedition and needed supplies, another would have the expedition and possible suppliers, and the last would have the supplier and the supplier's equipment.

Distributing a Form and Its Template

For a user to work with a form, its template and any associated database tables must be stored on a file server, a central location, or on the user's local drive. For example, templates can be stored in cc:Mail bulletin boards or Lotus Notes databases, or placed on a local area network.

To decrease network traffic, set up a form's routing list to send just the data. By keeping a form's data separate from its template, you decrease the amount of network time and space needed to route the form from stop to stop.



See "Distributing a Form" in the How Do I section of Help and Chapter 11 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more details on releasing forms to users.

Chapter 13

Working with the Sample Forms

cc:Mail Forms Designer has 12 sample form templates that are set up to run complete applications. For example, you can work with the phone message form to enter a phone message and route the message to the recipient.

This chapter includes the procedures you'll need to follow in order to explore the sample forms.

There are two approaches for working with these sample forms:

- You can run the sample form as a user, completing the form, routing it to the next stop, and then becoming the recipient at the next stop to receive and open it. For example, you can complete an expense report form and e-mail it to a supervisor for approval. You can then receive the expense report as the supervisor and approve or reject the form.
- You can look at the templates as a form developer, seeing how the database links are set up, where scripts are included, how routing lists are used, and how other form features are implemented.

This chapter consists of the following sections:

- Understanding the sample post office
- Opening sample form templates
- Viewing scripts
- Displaying routing lists
- Displaying database link windows
- Switching between Designer mode and Filler mode
- Closing a sample form template
- Getting Help for running the sample form

Understanding the Sample Post Office

During installation, a sample Lotus cc:Mail post office was installed, for use with the sample forms. This post office is located in the C:\DESIGNER\WORK\SAMPLES directory, or the directory you specified during installation.

The following table lists the names and passwords used in the sample post office.

<i>Name</i>	<i>Password</i>
Pam Ressler	PR
Shane DiCristina	SD
Matt Mai	MM
Liz Murray	LM
Jeff Falco	JF
Zac Blanchette	ZB

Opening a Sample Form Template

For hands-on exploration, you must open a sample template file. You can have several form templates open at a time. Template files have the .LTM file extension.

Note The following procedure assumes that the sample template files were installed in C:\DESIGNER\WORK\SAMPLES.



1. Display the Designer window.
2. Choose File - Open Form Template.
3. Under Directories, double-click Samples.
4. Under File name, select the template you want to open.
5. (Optional) To preview the contents of the selected template, click Preview. Click Preview again to close the Preview window.
6. Click OK.

The sample template appears in the Designer window. Other windows remain open.

Viewing Scripts

You can view scripts for layout objects and for the form itself. As you browse through the sample templates you'll find scripts that perform simple tasks, such as going to the next page when the user clicks a button, and complex tasks, such as writing information to a database, or linking to a database to specify the list of options in a field.

The scripts for a form determine what happens when Forms Filler creates, opens, saves, and closes a form, as well as any declarations for the form.

See Chapters 8, 9, and 10 and Appendix B in *Lotus cc:Mail Forms Designer Application Developer's Guide* for descriptions of other ways of viewing scripts for layout objects and forms.

To view a layout object's script

1. Position the mouse pointer on the object.
2. Right-click to display the Quick menu.
3. Select Script.

cc:Mail Forms Designer selects the object, switches to Script view, and displays the script for the selected object. As you're reviewing the script, also note that the LotusScript language identifies the object by its field name. This name also appears in the object's properties dialog box on the Basics panel.

To view a form's script

1. Click the Script tab to switch to Script view (in Designer mode).
2. Under Object, choose Form.
3. Under Proc, choose NewForm, OpenForm, SaveForm, CloseForm, or Declarations.

Displaying a Form's Routing List

A routing list shows the stops on the route and the roles used as stop recipients. A routing list is part of a template, and is created in the Routing view of the Designer window.

1. Open a sample template file.
2. Click the Routing tab to display Routing view.

The upper portion of Routing view displays the stops in the routing list. The lower portion displays icons representing the roles used in the routing list.

You can display the properties dialog box for a stop and a role just as you would for a layout object in Form Layout view. That is, you can double-click the stop or role to display the properties dialog box, or you can select the stop or role and then select Properties from the object's menu.

Similarly, you can display the script for a stop or role just as you would for a layout object. See the preceding section for more information.

See Chapter 7 in this book, and Chapter 6 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more about routing lists.

Displaying Database Link Windows

A database link window lists the fields in a database table. When a form field is linked to a database field, a line connects the field in the database link window to the form field.

To display database link windows

1. Open a template file that has database links (for example, PO.LTM).
Field names in red indicate fields that are linked to a database table through a database link window.
2. Choose View - Show - DBLink Windows.
All database link windows for the template appear. If there is more than one database link window, the windows may be stacked on top of each other.
3. Drag the title bar of the topmost database link window to reveal the next window in the stack.
4. Choose View - Show - DBLink Windows again to turn off display of the database link windows.

To display the properties dialog box for a database link window

1. Click the title bar of the database link window to select it.
2. Choose DBLink - Properties.

See Chapter 5 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information on database link windows.

Switching Between Designer Mode and Filler Mode

As you work, you will frequently switch between the Designer and Filler windows. You can do this quickly with SmartIcons.



- Click the Filler icon to go from the Designer to the Filler window.
- Click the Designer icon to go from the Filler to the Designer window.



The Filler window is a test environment. Any information or data that you enter in Filler mode is not saved when you return to Designer mode. If you want to save the data, use the File - Save command, in Filler mode, to create a form.

Closing a Sample Template

When you finish working in a sample template, it's a good idea to close it. Closing files you're not using saves memory and keeps your workspace from getting cluttered.



1. Make sure the file you want to close is the current template.
2. Choose File - Close, or double-click the control-menu box of the cc:Mail Forms Designer window.

If you changed the template but haven't saved the changes, you are asked whether to save the changes before closing.

- No ends the session without saving changes.
- Cancel returns you to cc:Mail Forms Designer without saving changes.

Getting Help for Running the Form Applications

Online Help displays information on all cc:Mail Forms Designer commands, dialog boxes, properties dialog boxes, and LotusScript commands. Help also provides information about each sample form, including how to run the form and which files you need to run it.

1. Choose Help - Contents or press **F1 (HELP)** to display the Help Contents window.
2. Click the Sample Forms icon.
3. Click the name of the sample form you want to run.
4. Scroll to the end of the Help topic.
5. Click the topic on filling out and sending the form.

The Help topic for running the form appears in the window.

6. (Optional) To print the Help topic, choose File - Print from the menu in the Help window.
7. (Optional) To keep the Help window on top of all other windows as you work through the procedure, choose Help - Always on Top from the menu in the Help window.

See Chapter 1 in *Lotus cc:Mail Forms Designer Application Developer's Guide* for more information about using Help.

Chapter 14

A Template Developer's Look at the Sample Forms

This chapter describes the features of the sample forms, emphasizing how a forms developer implemented each feature. The objective is to help you use the cc:Mail Forms Designer features to create the effects you want.

Because some features are conceptually the same from template to template, they're described just once. For example, all sample forms that e-mail a form use a routing list and roles database. On the other hand, because hiding and displaying fields at different stops on a routing list is implemented in several different ways across the sample forms, this chapter describes each method for hiding and displaying stops.

Twelve sample forms are installed with cc:Mail Forms Designer.

- The Phone Message form, the simplest form, demonstrates creating a three-dimensional design by placing layout objects on top of other objects, establishing a Dynamic Data Exchange (DDE) link with an e-mail system, using command buttons to e-mail a form, and using a command button to clear all data from the form.
- The Application for Employment, New Employee, Weekly Time Sheet, and Expense Report forms read and write information to the same database tables. When you run these forms, you can use the same applicant and employee for all four forms. This chapter shows how the template developer transferred information from one form to another, and set up and used the database links.
- The Request for Quotation (RFQ), Quotation Record, Purchase Order (PO), and Receiving Summary forms also read and write to the same database tables. You can see how the template developer created a system that sends out RFQs to vendors, uses an RFQ to create a Quotation Record, and then a Purchase Order. As the PO items are received, they are recorded in the Receiving Summary. This chapter shows how the developer set up the routing, determined which vendors receive which RFQs, and used database links and a tracking database.

- The Lead Response and Trip Reservation forms have trip information for a customer. You can record the customer's interests in the Lead Response form or click a button to create a Trip Reservation form and book a trip. This chapter shows how the developer used a form as a database record viewer, customized a menu command to create and display another form, and displayed fields based on the entry in another field.
- The Roles Database Editor is a database table editor, developed specifically for the cc:Mail Forms Designer roles database table. The developer created an application that is a database front-end that allows for quick and easy editing of the default roles database table.

See Chapter 13, for descriptions of the general procedures you use to open a sample form, display a script, view a database link window, and so on. Also, see *Sample Forms in Help* for information on running the sample forms and on the files you need to run them. See *Lotus cc:Mail Forms Designer Application Developer's Guide* and the *LotusScript Language Reference* for information on LotusScript commands and features.

Phone Message Form

To:	<input type="text"/>	Date:	<input type="text"/>
		Time:	<input type="text"/>
While You Were Out			
Mr./Ms.	<input type="text"/>		
Of	<input type="text"/>		
Phone	<input type="text"/>		
Telephoned	<input type="checkbox"/>	Would like to see you	<input type="checkbox"/>
Returned your call	<input type="checkbox"/>	Was in to see you	<input type="checkbox"/>
Will call again	<input type="checkbox"/>	Urgent	<input type="checkbox"/>
Please call	<input type="checkbox"/>	Sent you a FAX	<input type="checkbox"/>
Message <input type="text"/> <input type="text"/> <input type="text"/>		Send Clear	

The Phone Message form is an online rendition of the familiar pink message pad. This form records a message and e-mails it to the message recipient. When a user opens a new Phone Message form, Forms Filler fills in the date and time. The user then fills in the message information and clicks the Send button to e-mail the message to the recipient. To clear or erase all entries, the user clicks the Clear button.

Template file

The Phone Message form uses the template file MESSAGE.LTM. You can view the template in Designer mode and the finished form in Filler mode.

Feature highlights

The features implemented in the Phone Message form include examples of how to perform the following tasks:

- Create a three-dimensional effect by placing objects on top of rectangles
- Date- and time-stamp a form
- Create a combo box list using a pick list and a script
- Establish a DDE link to Lotus cc:Mail
- Clear, or erase, all information in a form

The following sections describe how the template developer implemented these features.

Creating a three-dimensional effect

You can create a three-dimensional effect in a form in different ways. One way is to choose a three-dimensional style for an object's frame. You select a style on the Style panel in the properties dialog box.

The developer of the Phone Message form created a three-dimensional backdrop by first drawing three rectangles and selecting a three-dimensional style. The developer next placed the combo box and text input fields (for example, To, Date, and Time) on top of the rectangles.

Date- and time-stamping a new form

When a form is created, Forms Filler executes the NewForm procedure for the form. In the NewForm procedure for the Phone Message form, the developer specified that Forms Filler enter the system date and time in the Date and Time fields, and place the focus on the To field. The following script excerpt from the NewForm procedure specifies these actions.

```
txtDate.Value=Date  
txtTime.Value=Format$(Time,"h:mm AM/PM")  
cmbTo.Edit
```

Creating a combo box list with a pick list and a script

You can create the list items for a combo box field in several ways. Two ways are to type the list items on the Pick List panel of the properties dialog box, or to specify the list items in a script. The developer for the Phone Message form used both of these techniques.

The Pick List panel specifies three names, and three are specified in the form's NewForm procedure. The following script excerpt shows how the names are added.

```
Dim MailList as listbox

Set MailList = cmbTo.listbox()

MailList.AddItem "Jeff Falco",3
MailList.AddItem "Shane DiCristina",4
MailList.AddItem "Zac Blanchette",5
```

Establishing a DDE link to e-mail

When a user clicks the Send button, Forms Filler creates a connection to cc:Mail and e-mails the message to the addressee in the To field. The message is sent in two formats: the form as it appears in cc:Mail Forms, and as an ASCII text message. The Send button's script executes these actions. The following is an excerpt from this script.

```
SUB BUTTONcmdSend (B1 AS BUTTON)

    Dim FormMail as DDE
    Dim f as Form
    Dim LF as String, MsgStr as String, AddStr as String
    LF = Chr(10)
    Set f=bind form("")
    If isnull (cmbTo.Value) Then
        Print "You must specify the name of an addressee."
    End
End if

AddStr = cmbTo.Value

Set FormMail = new DDE ("WMAIL.EXE","SendMail")
if FormMail is nothing THEN
    Print "You must start cc:Mail before sending."
End
END IF
```

```

FormMail.Execute("NewMessage")
    FormMail.Execute("Subject Messages from Telephone Center")

    FormMail.Execute("To " + AddStr)
    FormMail.Execute("Priority Urgent")

    FormMail.Execute("Text You received a message from the
        person below." + LF)
    FormMail.Execute("Text " +LF)
    FormMail.Execute("Text Name: ")

    If isnull (txtName.Value) = False Then
        MsgStr = txtName.Value
        FormMail.Execute("Text " + MsgStr + LF)
    Else
        FormMail.Execute("Text " +LF)
    End if

```

The script for the Send button also uses the Send method to e-mail the complete form. The argument for the Send method is the e-mail address, which is the name the user selected in the To combo box.

```

FormMail.Execute("Send")
FormMail.Terminate
f.Send(AddStr)

```

Clearing all data in a form

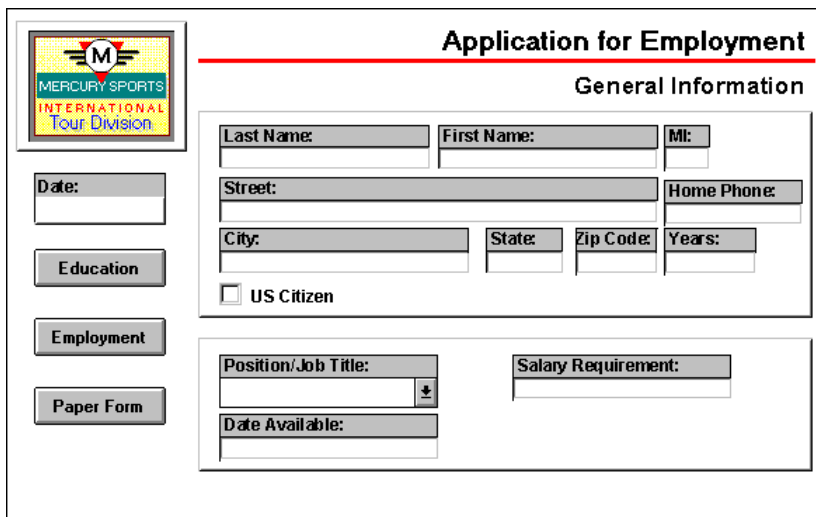
Clicking the Clear button erases all field entries in the Phone Message form. To accomplish this, the developer wrote the following script for the Clear button:

```

SUB BUTTONcmdClear (B1 AS BUTTON)
    Dim f as Form
    Set f=bind form("")
    f.Clear
END SUB

```

Application for Employment Form



Application for Employment

General Information

Mercury Sports International Tour Division

Date:

Education

Employment

Paper Form

Last Name: **First Name:** **MI:**

Street: **Home Phone:**

City: **State:** **Zip Code:** **Years:**

☐ US Citizen

Position/Job Title:

Salary Requirement:

Date Available:

The Human Resources department uses the Application for Employment form to record information on individuals applying for employment with Mercury Sports (a company used as an example in the sample forms). After completing the form, the recruiter clicks the Paper Form button to create a paper-based version of the form. Forms Filler transfers the information from the Application for Employment form to the paper-based form. This paper-based form can be e-mailed to the hiring manager for review and can be printed.

The Application for Employment form has three pages.

- Page 1 contains general information, such as the applicant's name, address, and salary requirements.
- Page 2 contains the applicant's education history.
- Page 3 contains the applicant's employment history.

When the recruiter saves the form, Forms Filler writes the information from each page to a corresponding database table.

Template files

The Application for Employment form uses two templates:

- The online form is used by the recruiter to enter the applicant information. This form uses the template file APP_EMPL.LTM.
- The paper-based form is printed and/or e-mailed to the hiring manager for review. This form uses the template file APP_2.LTM.

Mercury Sports originally used only the paper form. When they decided to enter the information online, they redesigned the form to make it easier to use on a computer.

You can view both templates in Designer mode, and use both forms in Filler mode.

Feature highlights

The features implemented in the Application for Employment form include examples of how to perform the following tasks:

- Use command buttons to navigate through the pages of a form
- Store information in database tables
- Use one form to open another form
- Transfer information from one form to another
- Print a form with a button
- Execute a menu command with a button
- Use a roles database

The following sections describe how the template developer implemented these features.

Using command buttons for navigation

In the online Application for Employment form, the Education, Employment, and General command buttons move the user to the corresponding pages of the form. The developer wrote a script for each button, and the script specifies the action that occurs when the user clicks the button. For example, the Education command button has the following script:

```
SUB BUTTONcmdEd_1 (B1 AS BUTTON)
    Dim Application as Form
    Set Application = bind form("")
    Application.gotopage 2
END SUB
```

The object library has buttons that already have a script to go to another page. To adapt these objects to suit your needs, change the number of the page to go to (in the script) and the button caption (in the properties dialog box).

Storing information in database tables

The Application for Employment form uses Structured Query Language (SQL) to store information in database tables. Forms Filler saves the information from each page of the form in a separate table. When the recruiter clicks the Paper Form button, Forms Filler first makes sure the required fields are filled in and then writes the information to an ODBC database, before opening the paper-based form.

The Paper Form button has the script that performs these actions. The following excerpt from this script writes the data from the General Information page to the database.

```
SqlStr(1) = "insert into app_gen(LName,FName,MI,Street,City,
    State,Zip,Phone,Years,Citizen,Position,Salary,ContOK)"

SqlStr(2) = "Values('"+ AddLN +"', '"+ AddFN +"', '"+ AddMI
    +"', '"+ AddSt +"', '"+ AddCity +"', '"+ AddState +"',
    '"+ AddZip +"', '"+ AddPhone +"', '"+ AddYrs +"', '"+
    AddCit +"', '"+ AddPos +"', '"+ AddSal +"', '"+ AddCont
    +'"')"
```

```
MstSql = ""

For i = 1 to 2
    MstSql = MstSql + SqlStr(i)
```

```

Next i

Set Applies = New db("cc:MailFormsApplicants")
Set App_S = Applies.ExecutesSql(MstSql)

    For Col = 1 to 5
    For Row = 1 to 4
    Set TmpField = tblEduc.Field(Row,Col)
    If TmpField.Value <>" " Then
        Educ(Row,Col) =TmpField.Value
    End if
    Next Row
    Next Col

```

Using one form to open another form

When the recruiter clicks the Paper Form button in the Application for Employment form, the information is written to the database tables and the paper-based version of the form is created. The script for the Paper Form button includes the following line, which creates the second form:

```
Set PaperForm = NewForm("App_2.ltm")
```

(The paper-based form uses the APP_2.LTM template.)

Transferring information from one form to another

When a recruiter clicks the Paper Form button, Forms Filler creates the paper-based form and executes its NewForm procedure script. This script transfers information from the online form to the paper-based form.

The following script is an excerpt from the NewForm procedure in APP_2.LTM. This script shows the commands executed to transfer the General Information data.

```

Set x = ScreenForm.Field("txtDate")
txtDate.Value = x.Value

Set x = ScreenForm.Field("txtFName")
Set y = ScreenForm.Field("txtMI")
Set z = ScreenForm.Field("txtLName")
txtName.Value = x.Value & " " & y.Value & " " & z.Value

Set x = ScreenForm.Field("txtStreet")
txtAddress.Value = x.Value

Set x = ScreenForm.Field("txtCity")
Set y = ScreenForm.Field("txtState")
Set z = ScreenForm.Field("txtZip")
txtAdd_2.Value = x.Value & " " & y.Value & " " & z.Value

```

```

Set x = ScreenForm.Field("txtYrs")
txtNOY.Value = x.Value

Set x = ScreenForm.Field("txtPhone")
txtPhone.Value = x.Value

Set x = ScreenForm.Field("chkCitizen")
If isnull(x.Value) Then
    txtCitizen.Value = "No"
Else
    txtCitizen.Value = "Yes"
End if

Set x = ScreenForm.Field("cmbPos")
txtPos.Value = x.Value

Set x = ScreenForm.Field("txtSal")
txtSalary.Value = x.Value

Set x = ScreenForm.Field("txtAvail")
txtAvail.Value = x.Value

```

Printing a form with a button

To print the paper-based form, the recruiter clicks the Print button. This button calls the Print method to print the form. PaperForm is the name of the form.

```

SUB BUTTONcmdPrint (B1 AS BUTTON)
    PaperForm.Print
END SUB

```

Executing a menu command with a button

To e-mail the paper-based form to the hiring manager, the recruiter clicks the Send button. The script for this button executes the Form - Send to Next Stop command.

```

SUB BUTTONcmdSend (B1 AS BUTTON)
    Form_SendtoNextStop.Execute
END SUB

```

Forms Filler routes the form, looking up the appropriate address in the roles database.

Using a roles database

When the recruiter e-mails the paper-based form, it is sent to a hiring manager. The position the applicant is applying for determines which hiring manager receives the form. The template developer set up a roles database that specifies the e-mail name of the hiring manager for each position. Forms Filler looks up the appropriate role in the database in order to route the form.

The following script for the Position role matches the value in the Position field with the appropriate hiring manager in the roles database.

```
SUB ROLEPosition (BYVAL UserName AS STRING, BYVAL RoleName AS
String)
Dim f as FORM
print "The current logged in user is " + username
SET f = BIND FORM("")
dim rolesdb as db
set rolesdb = new db("cc:MailFormsJobs")
dim rs as dbrecords

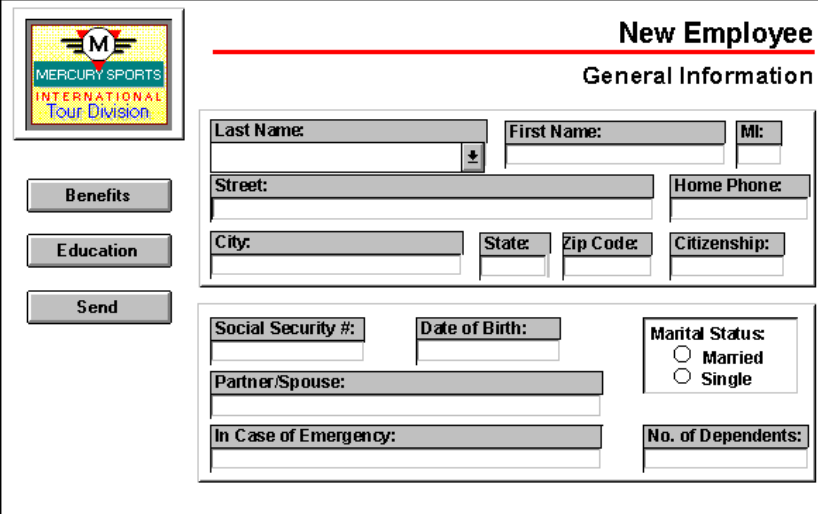
PosTitle = txtPos.Value

sqlexp$ = "select manager from jobs where position = '" +
PosTitle + "'"
set rs = rolesdb.executesql(sqlexp$)

if rs.count > 0 then
dim r as dbrecord
set r = rs.getrecord(0)
dim n as string
n = r.getattribute(0)
print "The address found for the role " + PosTitle + " is " +
n
f.RoleAddress = n
else
print "Unable to find Role to Route form"
end if

END SUB
```

New Employee Form



New Employee
General Information

Mercury Sports International Tour Division

Benefits
Education
Send

Last Name: [] First Name: [] MI: []
Street: [] Home Phone: []
City: [] State: [] Zip Code: [] Citizenship: []
Social Security #: [] Date of Birth: []
Partner/Spouse: []
In Case of Emergency: []
Marital Status:
☐ Married
☐ Single
No. of Dependents: []

When an applicant is hired at Mercury Sports, a Human Resources recruiter creates a New Employee form. The recruiter selects the new employee's last name from the list of applicants, and Forms Filler automatically fills in the fields with data retrieved from the applicant database tables. The recruiter then fills in additional information not included in the application for employment (such as marital status, date of birth, and benefits information). When the recruiter saves the form, Forms Filler writes the new information to a database. The recruiter next clicks the Send button, and the form is sent to Facilities Management where the facilities manager enters information on office space and computer equipment. Forms Filler then routes the form to Accounting.

The New Employee form has three pages:

- Page one contains general information, such as the employee's name, address, social security number, and date of birth.
- Page two contains the benefits information including the employee's start date, department, and insurance coverage.
- Page three lists the employee's educational history.

Template file



The New Employee form uses the template file NEW_EMPL.LTM. You can view the template in Designer mode and use the form in Filler mode.

See Sample Forms in Help for information on running this sample form and on the files you need to run it.

Feature highlights

The features implemented in the New Employee form include examples of how to perform the following tasks:

- Read and write information to and from database tables
- Set up combo boxes in tables
- Display hidden fields at specific stops

The next sections describe how the developer implemented these features.

Reading and writing data to and from database tables

The Application for Employment form writes information on all job applicants to the applicant database. When an applicant is hired and the recruiter selects the new employee's last name, Forms Filler queries the applicant database for that name and fills in the New Employee form with the existing information. Any new information the recruiter adds to the New Employee form is written to the appropriate database table.

To implement these features, the developer created these database links:

- The Last Name field is a combo box that displays a list of all last names in the applicant database table (APP_GEN). This link is made through the properties dialog box. The developer specified the database information on the Pick List panel of the properties dialog box for the field body.
- Three database tables link to the form through database link windows: APP_GEN, APP_EDUC, and PERSONAL. The Last Name field is the key field in all three tables. The properties for all three database link windows include Automatic Query. The form automatically writes data to the PERSONAL table, because the PERSONAL table is the only table that has the Read/Write property. (The other two tables have the Read Only property.)

Because the Last Name field is the key field for all three tables, when a user selects a last name from the pick list, Forms Filler automatically fills in the remaining linked fields from all three database tables.

- The Data panel in the properties dialog box for the form specifies that the information is written to the database when a user saves the form.

Using combo boxes in tables

On the Benefits Summary page (page two of the form), the developer used combo boxes that list the available options. This was done by selecting the field frame for column B, and, on the Style panel of the properties dialog box, setting the Object Type to Combo Box. The developer then entered the list choices in the Pick List panel of the properties dialog box for the field body of each cell.

Displaying fields at specific stops

The Office No. and Hardware fields, on the second page of the form, appear in the form only at the Facilities and Accounting stops. The template developer used the NewForm script to hide these fields when the recruiter creates the form.


The following script is the form's NewForm procedure:

```
SUB FormNewScript (f1 AS FORM)
    LName.Edit
    cmbOffice.Visible=False
    cmbHard.Visible=False
END SUB
```

To make the fields visible for the Facilities and Accounting mail stops, the template developer wrote a pre-processing script for each stop. Forms Filler executes a pre-processing script before opening a form at a stop. The Facilities stop has the following pre-processing script.

```
SUB PREROUTESTOPStop2 ()
    cmbOffice.Visible = True
    cmbHard.Visible = True
END SUB
```

Weekly Time Sheet Form

		Time Sheet																																																																			
	First:	Last:		Department:																																																																	
	Name:																																																																				
	SS #:			Week Ending:																																																																	
<table border="1"><thead><tr><th></th><th>Regular</th><th>O.T.</th><th>Vacation</th><th>Sick</th><th>Holiday</th><th>Total</th></tr></thead><tbody><tr><td>Monday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Tuesday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Wednesday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Thursday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Friday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Saturday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Sunday</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Total</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>								Regular	O.T.	Vacation	Sick	Holiday	Total	Monday							Tuesday							Wednesday							Thursday							Friday							Saturday							Sunday							Total						
	Regular	O.T.	Vacation	Sick	Holiday	Total																																																															
Monday																																																																					
Tuesday																																																																					
Wednesday																																																																					
Thursday																																																																					
Friday																																																																					
Saturday																																																																					
Sunday																																																																					
Total																																																																					
Employee's Signature:				<input type="button" value="Send"/>																																																																	

Employees in the sample company use the Weekly Time Sheet form to log the number of hours worked each week. When an employee clicks the Send button, the form is routed to the employee's payroll supervisor and then to the Accounting department. A Lotus Notes database tracks the progress of the form through the routing list. If a time sheet is delayed at a payroll approver's stop, the Notes database has a macro that automatically e-mails a reminder to the payroll approver to sign and mail the time sheet.

The form also writes information to a database table. The developer could design another form that uses this database information.

Template file

The Weekly Time Sheet form uses the template file TIME.LTM. You can view the template in Designer mode and use the form in Filler mode.

Feature highlights

The features implemented in the Weekly Time Sheet form include examples of how to perform the following tasks:

- Total table rows and columns.
- Track a form's progress through a route.
- Display a field at specific stops.

The following sections describe how the template developer implemented these features.

Totaling table and row columns

As an employee enters hours in the time sheet, the form automatically totals the hours for each day, and totals the number of regular, overtime, vacation, sick, and holiday hours. The developer implemented this by selecting the Row Totals and Column Totals options on the Basics panel of the properties dialog box for the table.

Tracking a form's progress through a route

To track a form's progress, you specify a database table, the fields you want written to the table, and at which stops you want the tracking database updated with the field information. The tracking database and fields are specified using the Route - Tracking command in the Routing view. To write the information from a stop, select the Update Tracking Database option on the Tracking panel of the properties dialog box for the stop. The form's location can be determined by querying the tracking database.

For the Weekly Time Sheet form, the developer specified a Lotus Notes database both as the tracking database and as the database that provides the view of the tracking status in Notes. The Notes database also contains a macro that determines when a time sheet is being held at a supervisor's stop, and e-mails a reminder to the supervisor to approve and e-mail the delinquent time sheets.

The fields that write information to the tracking database are the RouteStop, FromName, and ToName fields.

In addition to the fields that are tracked, the developer uses two hidden fields to write information to the database. These two fields, txtPayrollApprover and txtLocation, are at the bottom of the form.

- The txtPayrollApprover field holds the e-mail name for the corresponding role.
- The txtLocation field holds the role name of who has the form.

The txtLocation field information appears in the tracking database. The Lotus Notes macro checks the location, and if the location is the Payroll Approver and the form is delayed, a reminder notice is sent to the e-mail name in the txtPayrollApprover field.

Forms Filler writes the location and e-mail name to these fields when the user clicks the Send button. The form is then routed to the next stop.

Clicking the Send button executes the following script.

```
SUB BUTTONbtnSend (B1 AS BUTTON)

if (ISNULL(txtFirstName.VALUE)) OR
  (ISNULL(txtLastName.VALUE)) OR
  (ISNULL(txtDepartment.VALUE))=TRUE THEN

Print "Make certain that a first name, last name, and department
      have been entered."

else
  SELECT CASE txtLocation.Value
    CASE "Employee"
      txtLocation.Value="PayrollApprover"    'store location
      'in invisible field
    CASE "PayrollApprover"
      txtLocation.Value="Accounting"
  END SELECT

Form_SendtoNextStop.Execute
end if

END SUB
```

Displaying a field at a specific stop

When an employee creates a Weekly Time Sheet form, the Supervisor's Signature field is hidden and disabled. To set these field properties, the developer deselected the Enabled and Visible options on the Format panel in the properties dialog box for the field's frame.


However, the Supervisor's Signature field must be visible at both the Supervisor and Accounting stops, and it must be enabled at the Supervisor's stop (so the supervisor can sign the form). To do this, the developer wrote pre-processing scripts to change the state of the field for each stop. A stop's pre-processing script is executed before the form is opened at that stop.

The Supervisor's stop uses the following pre-processing script to change the Supervisor's signature field to be enabled and visible:

```
SUB PREROUTESTOPSupervisor ( )
    sigSupervisor.Enabled=True
    sigSupervisor.Visible = True
END SUB
```

In the pre-processing script for the Accounting stop, the signature is changed to visible, but it is not enabled. Because the Accounting stop is the last route stop, its pre-processing script also hides and disables the Send button.

Expense Report Form

		Purpose					Explanation
		<input type="checkbox"/> Seminars/Professional Meetings <input type="checkbox"/> Adventure Facility Visit <input type="checkbox"/> Adventure Site Acquisition <input type="checkbox"/> Equipment Purchasing <input type="checkbox"/> Equipment Damage Appraisal <input type="checkbox"/> Staff Training <input type="checkbox"/> Staff Recruiting <input type="checkbox"/> Site Planning <input type="checkbox"/> Department Meeting					
Emp. No. _____	Dept. No. _____						
Last Name _____							
First Name _____							
Location _____							
Ending _____	Tel. Ext. _____						
Date	Description	Misc	Airline Tickets	Transportation	Hotel pkg&tips	Breakfast	L

Employees in the company use the Expense Report form to submit reimbursable expenses to Accounting. After an employee enters his or her employee number, the form queries a database to fill in the remaining employee information. As the employee fills out the form, the form automatically totals the rows and columns. When finished, the employee clicks the Send button to e-mail the form to the employee's supervisor. The supervisor signs the form, then clicks the Send button to e-mail the form to Accounting.

As in the Weekly Time Sheet form, the Approval signature field is hidden and disabled when an employee creates the Expense Report form. The field is visible at both the Supervisor and Accounting stops, and is enabled at the Supervisor stop so the supervisor can sign the form.

Template file

The Expense Report form uses the template file EXPENSE.LTM. You can view the template in the Designer and the form in the Filler.

Feature highlights

The features implemented in the Expense Report form include examples of how to perform the following tasks:

- Query a database table automatically.
- Total and subtotal table rows and columns.

The next sections describe how the developer implemented these features.

Querying a database table automatically

When an employee fills in the employee number, Forms Filler queries the database and fills in the remaining employee information. The employee information fields (employee number, department number, name, and phone extension) are linked, using a database link window, to the EMPLOYEE database table. This link is established with the Automatic Query setting on the Options panel in the properties dialog box for the database link.

The template developer specified the database link as a Read/Write link on the Options panel. This means that any changes the employee makes to the data in the linked fields is written to the database table.


The Write DB on Save option on the Data panel of the properties dialog box for the form specifies that the form writes information from linked fields to the database when the user saves the form.

Totaling and subtotalling table rows and columns

The Balance Due, Totals, Travel & Lodging Total, and Meals Total fields calculate subtotals. The developer wrote a script for these fields using the Formula procedure. For example, the Balance Due field uses the following script:

```
SUB FORMULAtxtBal (f1 AS FIELD)
    txtBal.Value = txtSum.Value - txtAdv.Value
END SUB
```

Request for Quotation Form

Request for Quotation				
		Mercury Sports, Intl. Tour Division 55 First Street Cambridge, MA 02142		
		Date of Request: <input type="text"/>		
		Reply By: <input type="text"/>		
		Ref. Number: <input type="text"/>		
Item #	Quantity	Description	Unit Price	Subtotal
			Total:	
<input type="button" value="Next Page"/>				

The Purchasing department uses the Request for Quotation form to provide vendors with information on products Mercury Sports wants to purchase. The form is partially completed by the Purchasing department and then routed to vendors for quotations. The vendors are integrated into the Mercury Sports computer system and can receive electronic mail forms. The vendors enter the necessary information into the form, and e-mail it back to the purchasing agent.

Forms Filler writes both the purchasing agent's and vendors' entries to the QUOTES.DB database. This information can then be transferred to a Quotation Record form (the next form described in this chapter).

The Request for Quotation form includes two pages:

- The first page lists the items for which the purchasing agent wants a quote. Vendors use this page to fill in the pricing information.
- The second page has the shipping and payment information. The purchasing agent also uses this page to select the vendors who will receive the Request for Quotation. The vendor information is not visible when a vendor receives the form.

Template file

The Request for Quotation form uses the template file RFQ.LTM. You can view the template in Designer mode and try it in Filler mode.

Feature highlights

The features implemented in the Request for Quotation form include examples of how to perform the following tasks:

- Link a pick list to a database.
- Move information from one form list to another list.
- Use a script to route a form.
- Write information to a database.
- Hide fields at a specific stop.

The next sections describe how the developer implemented these features.

Linking a pick list to a database

On the second page of the form, the List Vendors by Product combo box lists product categories. The purchasing agent selects one product category, such as Aviation, Food, or Kayaks. Forms Filler then executes the following script, which uses the product value to query VENDORS.DB for a list of appropriate vendors, and displays those vendors in the Vendors list box.

```
SUB AFTERlstProducts (f1 AS FIELD)

' Update the vendors list based on the product.

' Data Source Variables :
Dim dsSource as DB
Dim dsTable as DBRECORDS
Dim dsRecord as DBRECORD
Dim dsRecordIndex%
Dim lst as Listbox
Dim acProduct$
Dim acVendor$
Dim i%

acProduct = lstProducts.Value
set dsSource = new db("cc:MailFormsVendors")
if acProduct="All" then
    set dsTable = dsSource.executesql("select * from
vendors")
else
    set dsTable = dsSource.executesql("select * from vendors
where VendorProduct1='"+acProduct+"'")
end if
```

```

if dsTable.Count>0 then
  set lst=lstVendors.listbox()
  lst.Reset 'clear the current list
  for i=0 to dsTable.Count-1 'put the vendors in the
    'listbox
    set dsRecord=dsTable.GetRecord(i)
    acVendor=dsRecord.GetAttribute(FIELDVENDORNAME)
    lst.AddItem acVendor, i
  next i
end if

END SUB

```

The developer wrote the script for the AfterEdit procedure because the script must have a value, and thus can be run only after a user enters a value in the List Vendors by Product field.

Moving information from one form list to another list

The purchasing agent determines which vendors will receive the Request for Quotation on the second page of the form. The purchasing agent selects a vendor from the Vendors list and then clicks the Include button to add that vendor to the Mail List. The purchasing agent can also use the Remove button to take the selected vendor off the Mail List.

To accomplish these tasks, the developer wrote the following two scripts:

- The script for the general procedure MoveFromListToList moves a list item from one list to another.
- The scripts for the Include and the Remove buttons determine which vendor is being moved and which list (the Vendors list or the Mail List) is the source list and the destination list. The button scripts then call the MoveFromListToList procedure to move the vendor to another list.

The following script is the general procedure MoveFromListToList:

```
SUB MoveFromListToList (SrcList as Listbox, DestList as
Listbox)

    Dim acItem$
    Dim bOKToMove%
    Dim i%

    ' If an item is selected in the source list, try to move it
    ' to the destination list.

    if SrcList.ListIndex>-1 then
        acItem=SrcList.Value(SrcList.ListIndex)
        'Don't move it if it's already in the destination list.
        SrcList.RemoveItem SrcList.ListIndex    'go ahead and
            'remove from source
        bOKToMove=1    'assume it's not in the destination list.

        for i=0 to DestList.ListCount-1    'check to see whether
            'it is in list
            if acItem=DestList.Value(i) then
                bOKToMove=0
            end if
        next i
        if bOKToMove=1 then
            DestList.AddItem acItem, 0
        end if
    end if
END SUB
```

This next script is used by the Include button to move a vendor from the Vendors list to the Mail List:

```
SUB BUTTONbtnInclude (B1 AS BUTTON)

    ' Move vendor name in mail list.
    Dim lst1 as Listbox, lst2 as Listbox
    set lst1=lstVendors.Listbox() : set
        lst2=lstMailList.Listbox()
    MoveFromListToList lst1, lst2
END SUB
```

Using a script to route a form

The form's vendor distribution list is dynamically created as the purchasing agent adds vendors to the mailing list. When the purchasing agent or a vendor clicks the Send button, Forms Filler checks the "location" of the form to determine whether to mail the form to the vendors or to the purchasing agent.

The form specifies the location by one of the hidden fields (Vendor, Status, and Location) at the bottom of the second page. The Location field has the default value of Originator, when the form is created. (The Originator is the purchasing agent.) When the Location is the Originator, Forms Filler e-mails the form to the vendors and changes the Location to Vendor. Conversely, when the Location is Vendor, Forms Filler e-mails the form back to the purchasing agent and resets the Location to Originator.

The script for the Send button also writes the vendor name to the Vendor field and the status (Request or Response) to the Status field. If the form is being sent by the purchasing agent (the Originator) to a vendor, the status is Request; if the form is being sent by a vendor to a purchasing agent, the status is Response.

When a user saves the form, the information from the Vendor, Status, and Location fields is written to the QUOTES.DB database. The template developer uses the information from these fields for the Quotation Record form (the next form described in this chapter).

When a user clicks the Send button, Forms Filler executes the following script:

```
SUB BUTTONbtnSend (B1 AS BUTTON)

' Sends this form to each of the vendors in the mail list.

Dim frm as Form, lst as Listbox
Dim ndx%, cnt%
Dim acAddress$

set frm=bind form("")

SELECT CASE Location.Value

CASE "Originator"      'send form to the vendor
    Location.Value="Vendor"      'update status for form
that
    'is going to vendor(s)
    Status.Value="Request"      'anything going to the
```

```

vendor
    'is a request
    set lst=lstMailList.ListBox()
    cnt= lst.ListCount-1
    for i=0 to cnt
        acAddress=lst.Value(i)    'put the list value in a
        'string variable
        VendorName.Value=acAddress
        frm.send(acAddress)
    next i
    Location.Value="Originator"    'local form is at
Mercury

    CASE "Vendor"    'return form to Mercury
        Location.Value="Originator"
        Status.Value="Response"    'so Mercury will receive
        'form with this status
        frm.send("Matt Mai")
        Location.Value="Vendor"

    END SELECT

END SUB

```

Writing information to a database

The Request for Quotation form has a link to the QUOTES.DB database. Information is written to the QUOTES.DB database when the purchasing agent and the vendors save the form.

The template developer created the database link with a database link window. The template developer also specified that Forms Filler write information to the database when the form is saved. This property is set in the properties dialog box for the form, on the Data panel.

Hiding fields at a specific stop

When a vendor receives a Request for Quotation form, the section of the form (on page 2) where the purchasing agent selected the vendor is hidden. The template developer set this feature in the script for the form's OpenForm procedure. The following script executes when the vendor opens the form after receiving it from the purchasing agent:

```
SUB FormOpenScript (f1 AS FORM)

' Control enabled/visibility property of objects based on
the
' form's location:

    dim bFlag

    SELECT CASE Location.Value
        CASE "Originator"
            bFlag=TRUE
        CASE "Vendor"
            bFlag=FALSE
    END SELECT

' Specs:

    ReplyDate.Enabled=bFlag
    Item1.Enabled=bFlag : Quantity1.Enabled=bFlag
        :Description1.Enabled=bFlag
    Item2.Enabled=bFlag : Quantity2.Enabled=bFlag
        :Description2.Enabled=bFlag
    Item3.Enabled=bFlag : Quantity3.Enabled=bFlag
        :Description3.Enabled=bFlag
    Item4.Enabled=bFlag : Quantity4.Enabled=bFlag
        :Description4.Enabled=bFlag

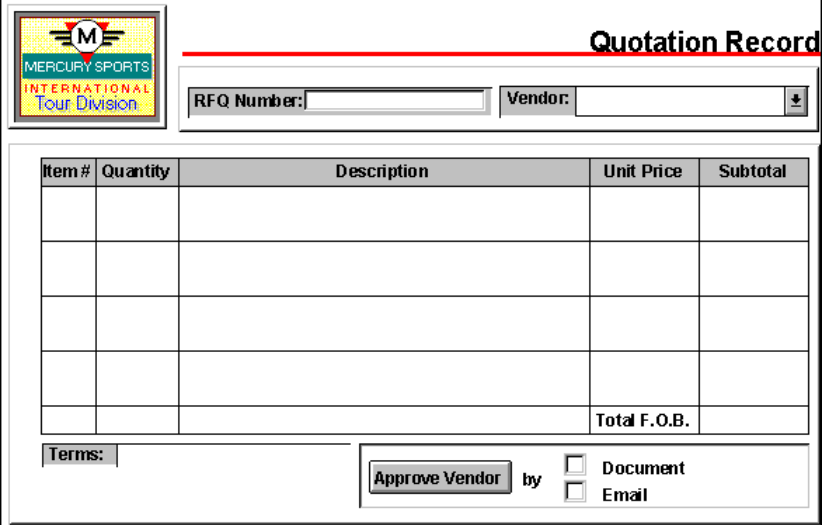
' Mail List Objects:

    grpVendors.Visible=bFlag
    lstProducts.Visible=bFlag
    lstVendors.Visible=bFlag
    lstMailList.Visible=bFlag
    btnInclude.Visible=bFlag
    btnRemove.Visible=bFlag

END SUB
```

Forms Filler executes an OpenForm script when a user opens an existing form. This includes opening a form received through e-mail or opening a form embedded in another document or application.

Quotation Record Form



The Quotation Record form is a Lotus Forms application. It features a logo for 'MERCURY SPORTS INTERNATIONAL Tour Division' in the top left corner. The title 'Quotation Record' is displayed in the top right. Below the title, there are two input fields: 'RFQ Number:' and 'Vendor:', each followed by a download icon. The main body of the form contains a table with five columns: 'Item #', 'Quantity', 'Description', 'Unit Price', and 'Subtotal'. The table has five data rows and a final row labeled 'Total F.O.B.'. At the bottom left, there is a 'Terms:' label followed by a text input field. At the bottom right, there is an 'Approve Vendor' button, a 'by' label, and two checkboxes labeled 'Document' and 'Email'.

Item #	Quantity	Description	Unit Price	Subtotal
			Total F.O.B.	

Terms:

by ☐ Document ☐ Email

The Purchasing department in the sample company uses the Quotation Record form to review quotes from vendors and to award the bid. A purchasing agent and vendors have already completed Request for Quotation (RFQ) forms, and cc:Mail Forms has written the information from the RFQs to the QUOTES.DB database.

To review the quotes, the purchasing agent queries the database for RFQs with vendor responses. cc:Mail Forms displays a list of the applicable RFQs, and the purchasing agent selects an RFQ to review. cc:Mail Forms transfers the RFQ information to the Quotation Record form. The purchasing agent can then approve a vendor and send that vendor a written confirmation using Lotus Word Pro® and/or cc:Mail. When the purchase agent saves the form, Forms Filler writes the linked fields to the PO.DB database table.

Template file

The Quotation Record form uses the template file QUOTES.LTM. You can view the template in Designer mode and the form in the Filler mode.

Feature highlights

The features implemented in the Quotation Record form application include examples of how to perform the following tasks:

- Specify a criterion to query a database.
- Integrate Word Pro and e-mail in a form application.

The following sections describe how the template developer implemented these features.

Specifying a criterion to query a database

To fill in a Quotation Record form, the purchasing agent chooses the Field - Query command and runs a query for RFQs with a status of Response (that is, RFQs with a vendor response). cc:Mail Forms displays a list of RFQs that meet the search criterion, and the purchasing agent selects an RFQ. Forms Filler then automatically reads the appropriate information from the QUOTES.DB and VENDOR.DB tables to fill in the Quotation Record form.

The setting to automatically query a database table is a database link property. This setting is on the Options panel of the properties dialog box for the database link window.

Integrating Word Pro and cc:Mail in a form application

The purchasing agent sends written approval to the vendor as a Word Pro document and/or an e-mail. To do this, the purchasing agent selects the appropriate check box(es) and then clicks the Approve Vendor button. The script for the Approve Vendor button checks the value of each check box and then creates the Word Pro document, e-mail message, or both.

Note Four hidden fields, at the bottom of the Quotation Record form, read the vendor's address from the VENDORS.DB table, and Forms Filler uses this information to address the Word Pro document.

The Approve Vendor button has the following script:

```
SUB BUTTONbtnApproveVendor (B1 AS BUTTON)
```

```
'   Notifies a vendor that it won the bid. Notification is by  
'   Ami Pro document and/or email.  
  
'   If a request for literature has been made, a message  
'   is sent to the Mail Department, instructing that the  
'   customer be sent the requested literature.  
  
'   DECLARATIONS AND INITIALIZATIONS
```

```

Dim LF as String
Dim TB as String
Dim acText$
Dim acList$
Dim msg$
Dim ddeMail as DDE
Dim ddeAmi as DDE
Dim ShellResult

LF= CHR(10)    'linefeed character

if ( ISNULL(VendorName.Value)=TRUE) OR
   (ISNULL(RFQNumber.Value)=TRUE) then
    MsgBox "Both an RFQ Number and a vendor must be
           specified."
    exit sub
end if

if ISNULL(chkDocument.Value)=FALSE then
ShellResult=Shell("amipro",7)  '7 is minimized without
focus
set ddeAmi=new DDE("amipro","SYSTEM")

ddeAmi.Execute
    "[fileopen(""c:\windows\template
developer\work\samples\
approve.asm"",1,""")]"

ddeAmi.Execute"[MarkBookMark(""VendorName"",4002)]"
    '4002 is FindBookMark
acText$=VendorName.Value 'put the value in a string
variable
ddeAmi.poke "VendorName",acText$    'poke the string

ddeAmi.Execute"[MarkBookMark(""VendorAddress"",4002)]"
acText$=VendorAddress.Value
ddeAmi.poke "VendorAddress",acText$

ddeAmi.Execute"[MarkBookMark(""VendorCity"",4002)]"
acText$=VendorCity.Value
ddeAmi.poke "VendorCity",acText$

ddeAmi.Execute"[MarkBookMark(""VendorState"",4002)]"
    '4002 is FindBookMark
acText$=VendorState.Value
ddeAmi.poke "VendorState",acText$

ddeAmi.Execute"[MarkBookMark(""VendorZip"",4002)]"
acText$=VendorZip.Value
ddeAmi.poke "VendorZip",acText$

```

```


ddeAmi.Execute"[MarkBookMark("RFQNumber",4002)]"
acText$=RFQNumber.Value
ddeAmi.poke "RFQNumber",acText$

ddeAmi.Terminate
end if

if ISNULL(chkEmail.Value)=FALSE then
set ddeMail=new dde("WMAIL.EXE","SendMail")
ddeMail.Execute("NewMessage")
ddeMail.Execute("Log 0")      'disable message logging
ddeMail.Execute("Receipt 0")  'disable the
                                'return-receipt
ddeMail.Execute("Subject Approval of Quotation")
acText=VendorName.Value
ddeMail.Execute("TO "+acText)
ddeMail.Execute("Priority Urgent")
msg$="We accept your quotation in response to RFQ
      #"+str(RFQNumber.Value)+". "
msg$=msg$+"You will receive a P.O. within the next few
      days."
ddeMail.Execute("Text "+msg$)
ddeMail.Execute("Send")
ddeMail.Terminate
end if
END SUB

```

Purchase Order Form

		Purchase Order	
		Date _____	P.O. Number _____
		R.F.Q. Number _____	Buyer's Name _____
Vendor		Shipping & Payment	
Company Name _____		Delivery Date _____	
Address _____		F.O.B. _____	
City _____	State _____ Zip _____	Ship VIA _____	
Attention (Name/Dept.) _____ Phone _____		Payment Terms _____	
Ship To:		Bill To: <input type="checkbox"/> Same as Ship To	
Company Name _____		Company Name _____	
Address _____		Address _____	

A purchasing agent in the sample company can fill out a Purchase Order (PO) form either from scratch or by querying the quotes database and having Forms Filler fill in the information. After signing the PO, the purchasing agent clicks the Send button to e-mail the form to the agent's approver. The form also writes the amount of the PO to a Lotus 1-2-3® worksheet.

POs that total \$10,000 or more require two approval signatures, and POs that total less than \$10,000 require only one approval signature. After the first approver signs and mails the PO, Forms Filler determines whether the PO must go to a second approver before being sent to the vendor.

Template file

The Purchase Order form uses the template file PO.LTM. You can view the template in Designer mode and the form in Filler mode.

Feature highlights

The features implemented in the Purchase Order form include examples of how to perform the following tasks:

- Enter default values in fields.
- Create a DDE link with Lotus 1-2-3.
- Conditionally route a form.
- Enable fields at specific stops.

The following sections describe how the template developer implemented these features.

Entering default values in fields

When the user views the form in the Filler, Forms Filler automatically fills in the values for the Ship To fields. The template developer entered these values in the properties dialog box for the field's frame, on the Basics panel. The purchasing agent can change these defaults by typing in a different address; however, these changes affect the current form only, they do not change the template.

Creating a DDE link with Lotus 1-2-3

When the purchasing agent sends the form to the first approver, Forms Filler also creates a DDE link with the 1-2-3 worksheet PO.WK4. Then Forms Filler writes the PO number, vendor name, and total to the worksheet. These actions are specified in the following post-processing script for the Purchase stop (the first stop in the routing list).

```
SUB POSTROUTESTOPPurchaser ()

' This procedure establishes a DDE conversation with 123R4W.
' It then sends data to a spreadsheet.

' Declare variables:
Dim dde123 as DDE
Dim t$ 'temporary text string

' Start 123 and the DDE conversation:
x=shell ("123W C:\DESIGNER\WORK\SAMPLES\PO.WK4",7)
'7 is minimized without focus
Set dde123 = new DDE
("123W","C:\DESIGNER\WORK\SAMPLES\po.wk4")

' This is an example of how you could get data from 123:
' txtFoo.Value=dde123.Request ("A:A1")
```

```

' Send data to spreadsheet by poking them into name cells:

t$=PONumber.Value
dde123.Poke "PONumber",t$
t$=VendorName.Value
dde123.Poke "VendorName",t$
t$=t_E6.Value 'this is the "Total" field of the table
dde123.Poke "Amount",t$

' End the DDE conversation:
dde123.Terminate

END SUB

```

Conditional routing of a form

The PO form uses conditional routing. After the first approver signs and mails the form, Forms Filler must determine whether the next stop is the second approver or the vendor. If the amount of the PO is \$10,000 or more, the next stop is the second approver; otherwise, the next stop is the vendor.

The post-processing script for the Approver1 stop determines where the form goes next.

```

SUB POSTROUTESTOPApprover1 ()

    Dim PO as Form
    Dim POVal as Single

    Set PO = bind form("")
    POVal = t_E6.Value

    If POVal <= 10000 Then
        PO.RouteStopName = "Vendor"
    End if

END SUB

```

The template developer used the Post-Process procedure because cc:Mail Forms Filler executes this script before sending the form to the next stop.

Enabling fields at specific stops

The PO form has three signature fields, which are displayed at all times. However, each signature field is enabled only at the appropriate stop. For example, the First Approval signature field is always displayed, but can be signed only at the Approver1 stop in the routing list. To enable this signature field, the template developer wrote the following script for the Pre-Process procedure of the Approver1 stop.

```

SUB PREROUTESTOPApprover1 ()
    sig1stApproval.Enabled=TRUE
END SUB


```

Because the purchasing agent is the first person to work in the form, the Buyer's signature field is enabled when the form is created. The following line of script, from the form's NewForm procedure, enables the Buyer's signature field.

```
sigBuyer.Enabled=TRUE
```

Receiving Summary Form

Receiving Summary
General Information



Date _____ **P.O. Number** _____

Time _____ **Inspector's Name** _____

Received From

Company Name _____

Address _____

City _____ **State** _____ **Zip** _____

Phone _____

Next Page

The Purchasing department of the sample company uses the Receiving Summary form to document the receipt of goods. Based on inspection of incoming merchandise, shipments are recorded as received or rejected. The form is routed from the receiving inspector to the appropriate supervisor, and the supervisor routes the form to the Accounting department.

The Receiving Summary form is embedded in a Lotus Notes document. The purchasing inspector, supervisor, and accountant create, route, and view the Receiving Summary form within Lotus Notes. The template developer used Notes/FX to transfer information from the embedded cc:Mail Forms object to the Notes document. The Lotus Notes document gives the user two views of the status of orders: by vendor and by status.

The Receiving Summary form has the following two pages:

- The first page displays the date and time, PO (purchase order) number, receiving inspector's name, and the vendor's name and address.
- The second page displays the items ordered, and the quantities ordered, received, and rejected.

Template file

The Receiving Summary form uses the template file RECEIVE.LTM. You can view the template in Designer mode and the form in Filler mode.

Feature highlights

The features implemented in the Receiving Summary form include examples of how to perform the following tasks:

- Exchange field data with Lotus Notes.
- Use a hidden field to determine a form's status.
- Initialize a form using both the NewForm and OpenForm procedures.

The following sections describe how the template developer implemented these features.

Exchanging field data with Lotus Notes

With Notes/FX a cc:Mail Forms document and a Lotus Notes document can exchange information. The Receiving Summary uses Notes/FX to display the current state of a PO in a Lotus Notes document. The information displayed in Notes is taken from three fields: the PO number, the vendor name, and the status of the PO.

Note The status field is a hidden field at the bottom of page two of the Receiving Summary form.

To have an exchange, both applications' documents must have the same fields. In the cc:Mail Forms document, the fields must also have the Notes/FX property. You set this property on the Format panel in the properties dialog box for the field's frame.

See Chapter 11 for detailed procedures that show how to set up field exchange between Forms Filler and Lotus Notes, and how to embed a form in a Notes document.

Using a hidden field to determine a form's status

The txtStatus field is a hidden field on the second page of the Receiving Summary form. The template developer uses this field to determine the status of the purchase order; that is, whether it is pending, incomplete, or complete. The script for this field calculates the status and returns the result to the field body. This field value is then read into the corresponding field in the Notes document.

The template developer wrote the following script for the Formula procedure of the txtStatus object.

```
SUB FORMULAtxtStatus (f1 AS FIELD)

    Dim a as field    'variables for table cell fields,
        'QuantityOrdered
    Dim b as field    'QuantityReceived
    Dim c as field    'QuantityRejected
    Dim bIncomplete%  'used to set txtStatus.Value

    Dim foo    'dummy variable used to trigger event
    Dim Row%
    Dim x%, y%, z%

    foo=TotalOrdered.Value    'just used to trigger this
formula
    'event
    foo=TotalReceived.Value    'just used to trigger this
formula
    'event
    foo=TotalRejected.Value    'just used to trigger this
formula
    'event

    bIncomplete=0    'init flag to 0

'   Loop through each row of the table:

    for Row=1 to 4
        set a=t.Field(Row,2)    'get Quantity Ordered
        set b=t.Field(Row,3)    'get Quantity Received
        set c=t.Field(Row,4)    'get Quantity Rejected
        if ISNULL(a.Value)=TRUE then x=0 else x=a.Value
        if ISNULL(b.Value)=TRUE then y=0 else y=b.Value
        if ISNULL(c.Value)=TRUE then z=0 else z=c.Value
        if x>(y-z) then bIncomplete=bIncomplete+1    'order has
            'not been fulfilled
    next Row
    if bIncomplete>0 then txtStatus.Value="Incomplete"
        else txtStatus.Value="Complete"

END SUB
```

Initializing a form

You can use the NewForm and OpenForm procedures to initialize a form. Forms Filler executes the NewForm procedure when a user creates a new form, and executes the OpenForm procedure when the user opens an existing form. The NewForm procedure is executed only once; the OpenForm procedure is executed every time a form is opened.

When you embed a form, cc:Mail Forms executes its NewForm script. When users then launch or open the embedded form, Forms Filler executes the OpenForm script.

The Receiving Summary form has both a NewForm and an OpenForm script. Both scripts set the time and date values in the form. This means that regardless of whether the form is being created or opened, it has the current time and date.

The NewForm script also enables the Inspector's signature; however, the OpenForm script does not. Because the NewForm script was run when the form was embedded, the Inspector's signature is always enabled when the Receiving Summary is opened from within Lotus Notes.

The template developer wrote the following script for the NewForm procedure:

```
SUB FormNewScript (f1 AS FORM)
' This procedure time-stamps the form.
  txtDate.Value = Date
  txtTime.Value=Time
  sigInspector.Enabled=TRUE
END SUB
```

This next script is the OpenForm procedure:

```
SUB FormOpenScript (f1 AS FORM)
' This procedure time-stamps the form.
  txtDate.Value = Date
  txtTime.Value=Time
END SUB
```

Lead Response Form

The screenshot shows a web form titled "Lead Response" with a red header bar. On the left is a logo for "MERCURY SPORTS INTERNATIONAL Tour Division" featuring a stylized 'M' with wings. The form contains several input fields: "Name:" (split into two boxes), "Interest:", "Address:", "City:", "State:", "Zip:", and "Phone:". Below these is a navigation bar with buttons labeled "K", "<", ">", "I", "Reservation...", and "Done". On the right side, under the heading "Requested Literature:", there is a list of ten activities, each preceded by an unchecked checkbox: Bungee Jumping, Hot-Air Ballooning, Jet-Fighter Ace, Kayaking, Mountaineering, Orienteering, Photo Safari, Scuba Diving, Skydiving, and Spelunking. The last item, "Wilderness Trek", is partially cut off at the bottom of the list.

The Telemarketing department of the sample company uses the Lead Response form to view records in the sales leads database. The telemarketer can display a potential customer's record, call the customer, and check off the literature the customer wants to receive. The telemarketer then clicks the Done button, and cc:Mail Forms creates an e-mail with the customer and literature information, and sends the e-mail to the Mail department. The Mail department sends the requested literature to the customer.

If a customer is interested in making a trip reservation, the telemarketer clicks the Reservation button to create a Trip Reservation form. Forms Filler transfers the customer information from the Lead Response form to the Trip Reservation form, and the telemarketer enters the trip information.

See the Trip Reservation form, described later in this chapter, for more information.

Template file

The Lead Response form uses the template file LEAD.LTM. You can view the template in Designer mode and the form in Filler mode.

Feature highlights

The features implemented in the Lead Response form include examples of how to perform the following tasks:

- Use LotusScript to connect to a database.
- Use buttons to display database records.
- Establish a DDE link to e-mail.

The following sections describe how the template developer implemented these features.

Using a script to connect to a database

When the telemarketer creates a Lead Response form, Forms Filler executes the form's NewForm procedure, part of which makes the connection to the sales lead database. The following script excerpt shows how this connection is made.

```
*** Establish connection to database *****
set dsSource = new db("cc:MailFormsLeads")
set dsTable = dsSource.executesql("select * from leads")
*** Display the first record *****
dsRecordIndex=0
DisplayRecord
```

Using buttons to display database records

The telemarketer uses the arrow buttons (in the lower-left of the form) to display database records. Depending on which button the telemarketer clicks, Forms Filler displays the first or last record, or the previous or next record.

The developer implemented this feature using the following two scripts:

- The first script is the general procedure DisplayRecord, which displays the current record.
- The second script is written for the arrow buttons. The script for the buttons determines which record is the current record, then calls the DisplayRecord procedure to display the current record.

The following script is the general procedure DisplayRecord:

```
SUB DisplayRecord ()

'Displays the current record. Assumes dsRecordIndex is valid.

    set dsRecord = dsTable.GetRecord(dsRecordIndex)
    txtFirstName.Value=dsRecord.GetAttribute(FIELDFIRSTNAME)
    txtLastName.Value=dsRecord.GetAttribute(FIELDLASTNAME)
    txtAddress.Value=dsRecord.GetAttribute(FIELDADDRESS)
    txtCity.Value=dsRecord.GetAttribute(FIELDCITY)
    txtState.Value=dsRecord.GetAttribute(FIELDSTATE)
    txtZip.Value=dsRecord.GetAttribute(FIELDZIP)
    txtPhone.Value=dsRecord.GetAttribute(FIELDPHONE)
    txtInterest.Value=dsRecord.GetAttribute(FIELDINTEREST)

    txtCount.Value=str(dsRecordIndex+1)+" of
        "+str(dsTable.Count)

END SUB
```

This next script is the Select procedure for the btnPgDn object (the next record button):

```
SUB BUTTONbtnPgDn (B1 AS BUTTON)

'Go to the next record.
if dsRecordIndex<dsTable.Count-1 then
    dsRecordIndex = dsRecordIndex+1
    DisplayRecord
else
    beep
end if

END SUB
```

Establishing a DDE link to e-mail

After checking off the literature the customer wants to receive, the telemarketer clicks the Done button. The script for this button creates an e-mail that contains the customer's address and requested literature, and sends the e-mail to the Mail department.

The e-mail connection is a DDE link that is established when the telemarketer creates a new Lead Response form. When creating the document, Forms Filler executes the form's NewForm procedure. The following line of script from this procedure establishes the DDE connection:


```
set ddel = new DDE("WMAIL.EXE","SendMail")
```

When the telemarketer closes a Lead Response form, Forms Filler executes the following CloseForm procedure for the Form object. This procedure terminates the DDE connection.

```
SUB FormCloseScript (f1 AS FORM)
    ddel.Terminate
END SUB
```

The e-mail link is in effect from the time a Lead Response form is opened until it is closed. The telemarketer can keep the form open for as long as necessary, going from lead to lead (database record to database record), and sending e-mail requests to the Mail department.

Trip Reservation Form

	Trip Reservation Customer Information																						
	Date:	Time:	Reference ID:																				
<table border="1"> <tr> <td colspan="2">First</td> <td colspan="2">Last</td> </tr> <tr> <td>Name:</td> <td></td> <td></td> <td></td> </tr> <tr> <td>Company:</td> <td colspan="3"></td> </tr> <tr> <td>Address:</td> <td colspan="3"></td> </tr> <tr> <td>City:</td> <td></td> <td>State:</td> <td>Zip:</td> </tr> </table>				First		Last		Name:				Company:				Address:				City:		State:	Zip:
First		Last																					
Name:																							
Company:																							
Address:																							
City:		State:	Zip:																				
<table border="1"> <tr> <td>Home Phone:</td> <td>WorkPhone:</td> <td>Fax:</td> </tr> <tr> <td></td> <td></td> <td></td> </tr> </table>				Home Phone:	WorkPhone:	Fax:																	
Home Phone:	WorkPhone:	Fax:																					

The Telemarketing department in the sample company uses the Trip Reservation form to book trip reservations. The telemarketer can enter all necessary trip information, including the date and location of the trip, and the method of payment.

The Trip Reservation form has the following three pages:

- The first page has customer information, such as the address, phone numbers, and fax number.
- The second page has the map of trip locations, and, for each location, the types of trips available, trip dates, and trip descriptions.
- The third page has billing information such as the trip cost, number of people going on the trip, and the method of payment.

Once the telemarketer fills out the form, the telemarketer can preview and then print a paper-based version of the form to send as a confirmation to the customer. The telemarketer also e-mails the trip reservation information to the Accounting and Scheduling departments.

The Trip Reservation form can be created either as a form in the Filler or from the Lead Response form. If the Trip Reservation form is created from the Lead Response form, the customer's name, address, and phone number are automatically entered into the form.

Template files

This application uses two forms:

- One form is used by the telemarketer to enter trip information. This form uses the template file TRIP.LTM.
- The second form (a paper-based form) is printed and sent to the customer. This form uses the template file CONFIRM.LTM.

You can view the templates in Designer mode and the forms in the Filler mode.

Feature highlights

The features implemented in the Trip Reservation form include examples of how to perform the following tasks:

- Transfer information from one form to another.
- Create a custom menu command.
- Display fields based on the state of another field.
- Route a form to more than one recipient at a stop.

The following sections describe how the template developer implemented these features.

Transferring information from one form to another

When the telemarketer runs this form application, Forms Filler transfers information from the Lead Response form to the Trip Reservation form, and from the Trip Reservation form to the Trip Reservation Confirmation form.

Forms Filler transfers information from a Lead Response form to a Trip Reservation form when the latter is created. Forms Filler checks if the Lead Response form is open. If it is open, Forms Filler copies the customer information from the lead form to the trip form, as shown in the following lines of script. The following script segment is from the NewForm procedure for the Trip Reservation form:

```
' If the LEADS form is open, transfer the customer
' information from it.

set frm=FindFormWithTemplate("LeadResponseForm")

if frm is Nothing then
else

    set fld=frm.Field("txtFirstName")
    txtFirstName.Value=fld.Value
    set fld=frm.Field("txtLastName")
    txtLastName.Value=fld.Value
    set fld=frm.Field("txtAddress")
    txtAddress.Value=fld.Value
    set fld=frm.Field("txtCity")
    txtCity.Value=fld.Value
    set fld=frm.Field("txtState")
    txtState.Value=fld.Value
    set fld=frm.Field("txtZip")
    txtZip.Value=fld.Value
    set fld=frm.Field("txtPhone")
    txtHomePhone.Value=fld.Value

end if

END SUB
```

The Trip Reservation Confirmation form is created when the telemarketer chooses the File - Preview Confirmation command. The developer created this command specifically for this application. The command is described in the next section.

Creating a custom menu command

When the telemarketer completes the Trip Reservation form, the telemarketer chooses the File - Preview Confirmation command. Forms Filler then creates a Trip Reservation Confirmation form, and displays the form with all the trip information entered. The confirmation is a paper-based version of the Trip Reservation form, and can be printed and sent to the customer.

The File - Preview Confirmation command is a custom menu command created by the form template developer. The following procedure describes how to view the script for this command.

To view the script for a custom menu command

1. Display the Menu tab (in the Designer window).
2. Under the Menu Items menu bar, choose File - Preview Confirmation.
3. Click the Edit Script button.

The script creates a confirmation form using the CONFIRM.LTM template, and directs Forms Filler to run the NewForm procedure. The NewForm procedure transfers information from the open Trip Reservation form to the new confirmation form.

Displaying fields based on the state of another field

The third page of the Trip Reservation form specifies the method of payment. If the telemarketer selects a credit card, Forms Filler displays three additional fields for the credit card's account number, and expiration month and year.

The developer created these fields with the following settings and script:

- The account number and expiration fields are enabled, but not visible. You set these properties on the Format panel of the properties dialog box for the object's field frame.
- The option buttons next to the credit cards have Select Values set to 0, 1, and 2 (for the first, second, and third credit card, respectively). You set these values on the Basics panel of the properties dialog box for the object's field body.
- The option buttons all have the same field name and script.

The following script specifies that if the user selects an option button with the value 0, 1, or 2, cc:Mail Forms displays the account number and expiration fields:

```
SUB AFTERobPaymentMethod (f1 AS FIELD)

' This procedure makes visible credit card account number and
' expiration fields if the method of payment is credit card
' (f1.value = 0,1,2).
  Dim bVisible 'flag used to set visibility
  if f1.Value=0 or f1.Value=1 or f1.Value=2 then
    bVisible=TRUE 'make them visible
  else
    bVisible=FALSE 'make them invisible
  end if
' Set the visibility of the relevant objects:
  grpAccountNumber.Visible = bVisible
  fldAccountNumber.Visible=bVisible
  lblExpirationDate.Visible=bVisible
  fldExpirationMonth.Visible=bVisible
  lblMonth.Visible=bVisible
  fldExpirationYear.Visible=bVisible
  lblYear.Visible=bVisible

END SUB
```

Because Forms Filler executes the script after the user selects an option button, the template developer used the AfterEdit procedure for the obPaymentMethod objects.

Routing a form to more than one recipient at a stop

The routing list for the Trip Reservation form has two stops, Telemarketing and Distribution. The Telemarketing stop is the Originator, and the Distribution stop has two recipients, Accounting and Scheduling. When the telemarketer chooses the Form - Send to Next Stop command, Forms Filler refers to the roles database to route the form simultaneously to the appropriate people in the Accounting and Scheduling departments.



See Help for information on routing to more than one recipient at a stop. Search for “stops” then select the topic “Details: Selecting a Stop Address.”

Roles Database Editor

The screenshot shows a form titled "cc:Mail Forms Roles Database Editor" with a teal background. In the top right corner, it says "Record 1 of 19". The form contains three text input fields: "Role:" with the value "Accounting", "From Name:" with the value "Pam Ressler", and "To Name:" with the value "Pam Ressler". To the right of these fields is a control panel with four buttons: "<" and ">" for navigation, "Home" and "End" for jumping to the first or last record, and "Clear Fields" for resetting the inputs. At the bottom, there are four buttons arranged in a 2x2 grid: "ADD Current Entry as a New Role", "DELETE Current Role", "UPDATE Current Role", and "DELETE ALL Roles from the Database".

When you're developing a routing list for a template, you typically use roles, which are defined in a roles database table. The Roles Database Editor form application helps you to quickly and easily edit a roles database table. You use this application to browse the records in the roles database table, and to add, change, and delete existing records. You enter the role name, from name, and to name, then click the appropriate button to have the Roles Database Editor update the database table.

When you create or open a Roles Database Editor form, cc:Mail Forms Designer connects the form to ROLES.DB and displays the first record in the table. ROLES.DB is the default database table. This is the database table that cc:Mail Forms automatically uses in a role's script, and is the table used by all sample form applications that have a routing list. The simplest way to create a roles database table is to add your roles to ROLES.DB.

Alternatively, you can modify the Roles Database Editor template to connect to a different database table. If you do this, remember to also set up the data source for the table.

Note When you experiment with the Roles Database Editor form, it's recommended that you *not* delete or modify any existing records in the ROLES.DB table. These roles are used by the sample form applications, and changing one of these roles will result in one or more of the sample applications not running correctly. For a list of the records used by the sample forms, see Chapter 7.

Template file

The Roles Database Editor uses the template file ROLES.LTM. You can view the template in Designer mode and the form in Filler mode.

See Chapter 7 for information on running this sample form application, and Sample Forms in Help.

Feature highlights

The features implemented in the Roles Database Editor form include examples of how to perform the following tasks:

- Connect to a database table and display the first record.
- Display message boxes.
- Display a record counter that shows where you are in a database table.
- Clear displayed data.
- Add and update database records.
- Delete records.

The following sections describe how the developer implemented these features.

Connecting to a database table and displaying the first record

Whether you create a Roles Database Editor form or open an existing one, cc:Mail Forms Designer connects to the roles database table and displays the first record. This is accomplished with two scripts: NewForm (or OpenForm) and the general procedure DisplayRecord.

The form's NewForm and OpenForm procedures both use the same script to connect to the database table and to call the general procedure that displays the first record.

```
SUB FormOpenScript (f1 AS FORM)

' Establish a new db link
SET Roles_DB = NEW DB (DATASOURCE)

' Send SQL query to ODBC driver and return DBRecords object
' containing all records in the database

sqlexp$ = "select * from " + TABLENAME
SET RolesTable = Roles_DB.executesql (sqlexp$)

' Display the first record
dsRecordIndex=0
DisplayRecord

END SUB
```

Note If you use a database table other than ROLES.DB, you would edit the Form Declarations script to connect to the correct database table.

DisplayRecord is a general procedure created by the template developer. The script for this procedure displays the current record based on the record index. The record index is updated when you browse through the records, or add or delete a record.

```
SUB DisplayRecord ()

' Display the record in position dsRecordIndex in the
database.
' Also display the current record index.

DIM TempRoles_DB AS DB
DIM TempRolesTable AS DBRECORDS

' Verify that dsRecordIndex is valid
SET TempRoles_DB = NEW DB (DATASOURCE)
sqlexp$ = "Select * from " + TABLENAME
SET TempRolesTable = TempRoles_DB.ExecutesQL(sqlexp$)

IF (TempRolesTable.Count > 0) AND (dsRecordIndex <=
    TempRolesTable.Count) THEN
    SET currentRecord = RolesTable.GetRecord (dsRecordIndex)

    FromName.Value= currentRecord.GetAttribute(1)
    ToName.Value=currentRecord.GetAttribute(3)
    RoleName.Value=currentRecord.getattribute(2)

    recordCount.Value="Record" + STR$(dsRecordIndex+1) + " of
" +
        LTRIM$(STR$(RolesTable.Count))
ELSE
    FromName.Value = NULL
    ToName.Value = NULL
    RoleName.Value = NULL
    recordCount.Value = "Database contains zero records."
END IF
END SUB
```

Displaying message boxes

The Roles Database Editor form uses message boxes in two instances:

- When you first create a Roles Database Editor form, it displays a message reminding you that this application uses the same roles database table as the other sample forms.
- When you make a change, the Roles Database Editor writes the change to the database table. Because you cannot undo a change, the form application confirms all updates and deletions. When you click the Update, Delete, or Delete All buttons, a message confirms the action.

When you create a new Roles Database Editor form, cc:Mail Forms executes the form's NewForm script. This script displays the message that reminds you that you're editing the default roles database table.

```
MessageBox "The Roles Database Editor modifies the" _  
           + " default roles database table," _  
ROLES.DB." _  
           + " This table is used by the sample  
form" _  
           + " applications. Before deleting any  
records," _  
           + " refer to the Help or documentation on  
the" _  
           + " Roles Database Editor.", 0
```

When you click the Update, Delete, or Delete All button, a message displays the role information being updated or deleted, and requests that you confirm the action. For example, when you click the Update button, the following script is executed.

```
confirm% = MessageBox ("The current role: " + CHR$(10) +  
CHR$(10) _  
           + "Role: " + oldRoleName + CHR$(10) + "From name:  
" +  
           oldFromName + CHR$(10) _  
           + "To name: " + oldToName + CHR$(10) + CHR$(10) _  
           + "will be replaced with the following:" +  
CHR$(10) +  
           CHR$(10) _  
           + "Role: " + rolename.value + CHR$(10) + "From  
name: " +  
           fromname.value _  
           + CHR$(10) + "To name: " + toname.value, 1)
```

Displaying a record counter

The record counter is updated whenever you display a different record, or add or delete a record. The text displayed in this field is defined by the following line in the general procedure DisplayRecord.

```
recordCount.Value="Record" + STR$(dsRecordIndex+1) + " of " +  
    LTRIM$(STR$(RolesTable.Count))
```

The scripts for the browse buttons and the Add, Delete, and Delete All buttons update the record index. The value of the record counter (recordCount field) is then updated accordingly.

Clearing displayed data

When you click the Clear Fields button, the form removes the data displayed in the Role, To Name, and From Name fields from the display (but not from the database table). To set these field values to null, the template developer wrote the following script for the Clear Fields button.

```
SUB BUTTONbtnClearFields (B1 AS BUTTON)  
    ' Clear all information from the display fields, and  
    ' place the focus in the Role field.  
    rolename.value = NULL  
    fromname.value = NULL  
    toname.value = NULL  
    rolename.edit  
END SUB
```

Adding and updating database records

The scripts for adding and updating database records are similar. For example, when you click the Add button, Forms Filler executes the following script. This script first checks that the information for the Role, To Name, and From Name fields is complete and that this record does not already exist in the database table. The script then creates a new keynumber for the record and adds the record. Finally, the script updates the record index.

```
SUB BUTTONbtnAddRole (B1 AS BUTTON)  
  
    ' Append a new record to the end of the database, using the  
    ' current  
    ' information in the display fields.  
  
    Dim TempRoles_DB AS DB  
    DIM TempRolesTable AS DBRECORDS  
    DIM TempRecord AS DBRECORD  
    DIM max_key AS INTEGER  
    DIM new_key AS VARIANT
```

```

SET TempRoles_DB = NEW DB (DATASOURCE)

' Verify that all information for new role has been provided
If IsNull(ToName.value) OR IsNull (FromName.value) OR IsNull
  (RoleName.value) Then
  MessageBox "Please make sure all the fields are filled
in", 0
ELSE
  ' Verify that the role does not already exist in the
  database
  sqlexp$ = "select " + KeyNameFld + " from " + TABLENAME +
" where
    " + FromNameFld + " = '" + fromname.value + _
    + "' and " + RoleNameFld + " = '" + rolename.value + "'
and " +
    ToNameFld + " = '" + toname.value + "'"
  SET TempRolesTable = TempRoles_DB.executesql(sqlexp$)

  IF TempRolesTable.Count > 0 THEN
    MessageBox "This role has already been defined in the
roles
        database: " + CHR$(10) + CHR$(10) + _
        "Role: " + rolename.value + CHR$(10) + "From
name: "
        + fromname.value + _
        CHR$(10) + "To name: " + toname.value, 0
  ELSE
    ' Get the maximum existing key number, and create a new
    (unique)
    ' key number
    sqlexp$ = "Select MAX ( " + KeyNameFld + " ) ""max_key""
from " +
    TABLENAME

    SET TempRolesTable = TempRoles_DB.executesql(sqlexp$)
    SET TempRecord = TempRolesTable.getrecord(0)
    new_key = TempRecord.getattribute(0)

    IF IsNull(new_key) THEN
      new_key = 1
    ELSE
      new_key = new_key + 1
    END IF
  
```

```

sqlexp$="Insert into " + TABLENAME _
+ " (" + KeyNameFld + "," + FromNameFld + "," +
RoleNameFld + "," +
ToNameFld _
+ ") values (" + new_key + "','" + fromname.value + "','" +
rolename.value + "','" + toname.value + "')"
SET RolesTable = Roles_DB.executesql(sqlexp$)

' Update the table to include newly inserted role
sqlexp$ = "select * from " + TABLENAME
set RolesTable = Roles_DB.executesql (sqlexp$)
' Update the current record index to point to the newly
inserted
' role. The role is inserted at the end.
dsrecordindex = RolesTable.Count - 1

' Redisplay with new role as current role
DisplayRecord

END IF
End If
END SUB

```

Deleting the current record

When you click the Delete Current Role button, Forms Filler executes the following script. The script first checks that the Role, To Name, and From Name fields are complete. The script next displays a message box to confirm that you want to delete the current record. When you confirm the deletion, the script deletes the record, and updates the record index.

```

SUB BUTTONbtnDeleteRole (B1 AS BUTTON)

SET Roles_DB = NEW DB (DATASOURCE)

' Verify that all information for role to be deleted has
been
' provided
IF IsNull(ToName.value) OR IsNull (FromName.value) OR
IsNull
RoleName.value) Then
MessageBox "All fields must be completed before role
can be
deleted.", 0

```

```

ELSE
    confirm% = MsgBox ("If you continue, the current
entry will
    be permanently deleted from the Roles database." + _
    CHR$(10) + CHR$(10) + "Do you want to continue?", 4)
    IF confirm% = 6 THEN
        ' Delete the current record from the database.
        sqlexp$ = "Delete from " + TABLENAME + " where " +
            FromNameFld + "=" + fromname.value _
            + "' and " + RoleNameFld + "=" +
rolename.value+" ' and "
            + ToNameFld + "=" + toname.value + "'
        SET RolesTable = Roles_DB.ExecutesSQL(sqlexp$)

        ' Refresh table
        sqlexp$ = "Select * from " + TABLENAME
        SET RolesTable = Roles_DB.ExecutesSQL(sqlexp$)

        ' Display previous record
        IF dsRecordIndex <> 0 THEN
            dsRecordIndex = dsRecordIndex - 1
        END IF
        DisplayRecord
    END IF
END IF
END SUB

```

Index

A

Aligning field labels, 45
APP_2.LTM file, 125
APP_EMPL.LTM file, 125
Application for Employment
 form, 124
Automatic queries of database
 tables, 131, 137
 overview, 83
 scripted, 83
 setting, 80, 83

B

Bulletin boards
 storing form templates on, 9, 111
Buttons. *See* Command buttons.

C

Calculating row and column
 totals, 134, 137
cc:Mail, 63
 bulletin boards for storing form
 templates, 9, 111
 DDE link to, 122, 146
 post office for sample forms, 114
Check boxes
 creating, 45
 selecting parts of, 23
Clearing data from forms, 123, 168
Combo boxes
 creating, 35
 in tables, 132
 pick list items, 122
 pick lists linked to database
 tables, 81, 131, 139
 Pick Lists panel in properties
 dialog box, 81

Command buttons
 creating, 52
 executing menu commands
 with, 128
 going to another page with, 126
 printing forms with, 128
 writing scripts for, 55
Commands. *See* Command buttons;
 Menu commands.
Conditional routing, 2, 74, 151
CONFIRM.LTM file, 160
Copying. *See also* Duplicating layout
 objects.
 layout objects, 40
 layout objects into object
 library, 60
 library objects into form
 templates, 59

D

Data
 clearing from forms, 123, 168
 exchanging, with
 Notes, 99, 152, 153
 moving, between fields, 140
 transferring between
 forms, 127, 138, 145, 156, 160
Database front-end
 applications, 156, 164.
 See also Roles Database Editor.
Database link windows, 76
 automatic queries for, 80
 creating, 76
 deleting links to, 79
 displaying, 79, 116
 examples of, 131, 137
 field object names and, 78
 hiding, 79
 linking layout objects using, 77
 modifying fields in, 79
 pointer shapes for, 78
 properties dialog box for, 77, 80

Database link windows (*cont.*)
 read/write properties for, 80
 specifying automatic
 queries for, 83
 transaction isolation level for, 80
 viewing sample data with, 80
Database links, 4
 creating, 75
 deleting, 79
 uses for, 84, 100, 109
Database tables
 adding records to, 168
 connecting to, with
 scripts, 157, 165
 counting records in, 168
 creating links to, 75
 deleting records from, 170
 guidelines for setting up, 110
 linking pick lists to, 81, 131, 139
 reading and writing
 to, 126, 131, 134, 137, 143
 uses for, 84, 100, 109
 viewing records in, 157, 165
Date stamping forms, 121
DDE links
 to cc:Mail, 122, 146
 to e-mail, 158
 to Lotus 1-2-3, 150
Defaults. *See also* Initializing forms.
 field values, 150
 form values, 150
 roles database tables, 65
 roles scripts, 69
 setting form value, 155
Delrina files, importing, 9
Designer window, 15, 17, 18
 Form Layout view, 19
 Menu view, 20, 92
 Routing view, 19, 64, 70, 71
 Script view, 20
 switching to, 13

D (Cont.)

- Displaying fields
 - based on state of other fields, 162
 - pre-processing script for, 132, 136
- Documentation
 - conventions, xi
- Documentation, Lotus cc:Mail Forms Designer, x
- Duplicating layout objects, 33, 47.
See also Copying.
- Dynamic Data Exchange. *See* DDE links.

E

- E-mail
 - DDE link to, 158
 - paths for forms, 63
 - supported, systems, 4, 63
- Editing field captions, labels, and static text, 46, 48
- Employee form, 130
- Employment form, 124
- Erasing data from forms, 123
- EXPENSE.LTM file, 137
- Expense Report form, 136

F

- Field exchange with Lotus Notes, 153
- Field labels, aligning, 45
- Fields, 21, 22
 - check boxes, 45
 - combo boxes, 35
 - command buttons, 52
 - copying, 40
 - default values for, 150
 - disabling, 135
 - displaying scripts for, 115
 - duplicating, 47
 - editing text in, 46, 48
 - enabling, 135
 - grouping, 8
 - hiding/displaying based on state of other fields, 162
 - moving data between, 140
 - name changes, 78
 - overview, 4
 - selecting, 22
 - selecting parts of, 23
 - static text fields, 41
 - text input fields, 38, 43, 49
 - tracking data in, 87

- Filler window, 13, 16, 17
- Form Layout view, 19
- Form object, displaying script for, 115
- Form templates. *See also* Forms.
 - closing, 117
 - components of, 3
 - default values in, 150, 155
 - definition, 3
 - design considerations, 106
 - distributing to users, 111
 - integrating multiple, 119
 - layout guidelines, 106
 - opening sample files, 114
 - searching for, 9
 - testing, 110

Forms

- clearing data in, 123
- conditional routing of, 2
- creating Word Pro documents in, 146
- date and time stamping of, 121
- definition, 3
- displaying message boxes in, 167
- distributing to users, 111
- erasing data in, 123
- exchanging data with Notes, 99, 152, 153
- initializing, 155
- monitoring workflow of, 2
- opening, 155
- opening one from another, 127
- paper-based, 106, 127, 160
- routing to multiple recipients at one stop, 163
- selecting, 22
- tracking through routing lists, 87, 134
- transferring data
 - between, 127, 138, 145, 156, 160
- writing scripts for, 54

G

- General procedures,
 - examples of, 140, 157, 166
- Grouping objects, 8

H

- Help, for installation, 12
- Help, contents of, x
- Hiding fields, 135
 - based on state of other fields, 162
 - when creating a form, 132
 - when opening a form, 144

I

- Icons, 17
- Initializing forms, 155
- Installing Lotus cc:Mail Forms Designer, 11

L

- Labels, aligning field, 45
- Layout objects, 22
 - 3-D rectangles, 32
 - check boxes, 45
 - combo boxes, 35
 - command buttons, 52
 - copying, 40
 - copying to object library, 60
 - definition, 4
 - displaying scripts for, 115
 - duplicating, 33, 47
 - editing text in, 46, 48
 - fields, 4, 21
 - grouping, 8
 - menus for, 17
 - name changes of, in database link windows, 78
 - non-field objects, 4, 21
 - selecting, 22
 - static text fields, 41
 - text input fields, 38, 43, 49
 - types of, 21
- LEAD.LTM file, 156
- Lead Response form, 156
- Library. *See* Object library.
- List boxes, moving data
 - between, 140. *See also* Pick lists.
- Lotus 1-2-3, DDE links to, 150

- Lotus cc:Mail Forms Designer
 - Designer window, 15
 - documentation, x
 - field exchange with Notes, 99
 - Filler window, 16
 - installing, 11
 - introduction, 1
 - new features in Release 2.0, 6
 - SmartIcons, 17
 - starting, 12
 - switching between Designer mode and Filler mode, 13
 - window components, 17

- Lotus Forms, 1

- Lotus Notes, 63
 - as a tracking database, 90, 134
 - field exchange with, 90, 99, 153
 - storing form templates, 9, 111

- Lotus Notes/FX, 90, 99, 153
 - launching cc:Mail Forms Designer, 104
 - setting up the form, 100
 - setting up the Notes database, 101
 - setting up the Notes form, 101, 104
 - setting up the Notes view, 103
- Lotus Notes/FX. *See also* Receiving Summary form.

M

- Mail systems, supported, 4, 63

- Menu commands
 - adding to menus, 93
 - ALT key sequences for, 93
 - customizing, 162
 - deleting, 95, 96
 - executing with command buttons, 128
 - object specific, 17
 - reinstating standard, 96
 - shortcuts, 17

- Menu view, 20, 92

- Menus
 - adding commands to, 93
 - ALT key sequences for, 93
 - customizing, 20, 92, 162
 - deleting commands on, 95
 - deleting pulldown, 96
 - Main menu, 17
 - quick, 22

- Menus (*cont.*)

- Quick menus, 21, 45
 - reinstating standard commands, 96
 - right mouse, 22
- MESSAGE.LTM file, 29, 121
- Message boxes in forms, 167
- Message form, 121
- Microsoft Mail, 63
- Moving data
 - between fields, 140
 - between forms, 127, 138, 145, 156, 160

N

- Navigating forms with command buttons, 126
- NEW_EMPL.LTM file, 131
- New Employee form, 130
- Notes. *See* Lotus Notes.

O

- Object library
 - categories in, 61
 - copying objects to/from, 59, 60
 - index, 59
 - overview, 6
 - scripts in, 59
 - viewing objects in, 58
- Object library window, 58
- Objects. *See* Layout objects
- ODBC databases, 4
- Open Database Connectivity (ODBC) databases, 4
- Opening forms, 155
 - date and time stamping when, 121
 - from other forms, 127
- Opening sample templates, 114
- Originator stops, 64, 72

P

- Pages
 - selecting, 22
 - specifying color of, 31
 - specifying size of, 31
- Panels, in properties dialog box, 27
- Paper-based forms, 106, 160
 - transferring data to, 127, 160
- Phone Message form, 121

- Pick lists, 81
 - in combo boxes, 122
 - linking to database tables, 81, 131, 139
 - panel in properties dialog box, 81
- Picture fields, Release 2.0
 - features for, 7
- PO.LTM file, 149
- PO form, 149
- Pointer shapes in database link windows, 78
- Post-processing scripts, 63, 74
 - for conditional routing, 151
 - for connecting to 1-2-3, 150
- Post office, for sample forms, 114
- Pre-processing scripts, 63, 74
 - displaying fields with, 132, 136
 - enabling fields with, 136, 151
- Printing forms, 128
- Process Next Form command, 8
- Properties dialog box
 - closing, 28
 - for combo boxes, 81
 - for database link windows, 77, 80
 - for roles, 70
 - for stops, 73, 90
 - Help button, 27
 - moving, 28
 - overview, 25
- Purchase Order form, 149

Q

- Queries, 83
 - automatic, for database link windows, 80, 83
 - automatic, of database tables, 131, 137
 - database table, 146
 - scripted, 83
 - user specified, 83
- Quick menus, 21, 22
 - selecting from, 45
- Quotation Record form, 145
- QUOTES.LTM file, 145

R

- RECEIVE.LTM file, 153
- Receiving Summary form, 152
- Rectangles, creating 3-D, 32
- Request for Quotation form, 138
- RFQ.LTM file, 139
- RFQ form, 138
- Right mouse menus, 22, 45
- Roles, 63, 64
 - adding to routing view, 70
 - default script for, 69
 - example script for, 129
 - overview, 5
 - properties dialog box for, 70
- ROLES.LTM file, 165
- Roles Database Editor, 66, 164
 - adding roles with, 68
 - browsing records with, 68
 - deleting roles with, 68
 - updating roles with, 68
- Roles database tables, 63, 64, 65.
 - See also* Roles Database Editor.
- Routing forms. *See also* Routing lists.
 - conditionally, 74, 151
 - supported mail systems for, 63
 - to and from vendors, 138
 - to multiple recipients at one stop, 163
 - using scripts for, 142
- Routing lists, 63. *See also* Stops.
 - adding roles to, 70
 - adding stops to, 71
 - conditional, 151
 - conditional routing, 74
 - displaying, 116
 - guidelines for developing, 109
 - Routing view, 64
- Routing view, 19, 64
 - adding roles to, 70

S

- Sample form applications
 - Roles Database Editor, 66
- Sample form templates
 - opening, 114
- Sample forms
 - Application for Employment form, 124
 - closing templates for, 117
 - displaying database link windows in, 116
 - displaying routing lists for, 116
 - examining the design of, 119
 - Expense Report form, 136
 - Help for, 118
 - Lead Response form, 156
 - New Employee form, 130
 - Phone Message form, 121
 - post office for, 114
 - Purchase Order form, 149
 - Quotation Record form, 145
 - Receiving Summary form, 152
 - Request for Quotation form, 138
 - Roles Database Editor form, 164
 - Trip Reservation form, 159
 - Weekly Time Sheet form, 133
- Sample post office, 114
- Script view, 20
- Scripting, examples
 - e-mailing forms, 123
 - adding items to pick lists, 122
 - adding records to database tables, 168
 - calculating totals, 137
 - clearing data, 168
 - clearing forms, 123
 - conditional routing, 151
 - connecting to 1-2-3, 150
 - connecting to cc:Mail, 122
 - connecting to database tables, 157, 165
 - counting records, 168
 - creating other forms, 127
 - creating Word Pro documents, 146

Scripting, examples (*cont.*)

- date and time stamping forms, 121, 155
- deleting database table records, 170
- determining form status, 142
- displaying database records, 158, 165, 166
- displaying fields, 132, 136, 163
 - displaying message boxes, 167
 - e-mailing forms, 128, 146, 158
 - enabling fields, 136, 151
 - general procedures, 141, 158, 166
 - going to another page, 126
 - hiding fields, 132, 144
 - linking pick list items, 139
 - looking up roles, 69, 129
 - moving data between fields, 141
 - printing forms, 128
 - specifying stop recipients, 142
 - tracking forms, 135, 142, 154
 - transferring data between forms, 127, 161
 - writing data to 1-2-3, 150
 - writing to a database table, 126
- Scripts. *See also* Post-processing scripts; Pre-processing scripts.
 - creating database links with, 75
 - default roles, 69
 - displaying sample forms, 115
 - documentation for, xi
 - examples of use, 5
 - for customized menus, 91
 - for database queries, 83
 - in object library, 59
 - writing, 54, 55
- Scrolling text input fields, 49
- Selecting
 - check box and option button captions, 23
 - check box and option buttons, 23
 - forms, 22
 - objects, 22
 - pages, 22
 - parts of fields, 22, 23

- Shortcuts, 17
- Signature fields
 - disabling, 135
 - enabling, 135, 151, 155
 - hiding, 135
 - security for, 108
- SmartIcons, 17
- Spell checker, 6
- Starting Lotus cc:Mail Forms
 - Designer, 12
- Static text fields, 41
- Status bar, 18
- Stop recipients, 71
 - multiple, 163
 - specified with scripts, 142
- Stops, 63, 64. *See also* Roles.
 - adding to routing lists, 71
 - displaying fields at, 132, 135
 - enabling fields at, 135, 151
 - enabling for tracking database tables, 89
 - labels for, 71
 - Originator, 64, 72
 - properties dialog box for, 73, 90

T

- Tables
 - combo boxes in, 132
 - totaling rows and columns in, 134, 137
- Tabs
 - in Designer window, 18
 - in properties dialog box, 27
- Templates. *See* Form templates; Forms.
- Testing forms, 110
- Text fields, creating static, 41
- Text input fields
 - creating, 38, 43
 - creating scrolling, 49
- Three-dimensional effects, 121
- TIME.LTM file, 133
- Time Sheet form, 133
- Time stamping forms, 121
- Totaling rows and columns, 134, 137
- Tracking database tables, 87
 - enabling stops for, 89
 - examples of, 90, 133, 134
 - setting up, 87
- Transaction isolation level for
 - database link windows, 80
- TRIP.LTM file, 160
- Trip Reservation form, 159

V

- Views
 - Form Layout view, 19
 - Menu view, 20, 92
 - Routing view, 19, 64, 70, 71
 - Script view, 20
- VIM-compliant mail systems, 4, 63

W

- Weekly Time Sheet form, 133
- Word Pro documents, created by forms, 146
- Workflow
 - monitoring forms, 87

Z

- Zoom percentages, 18