

## Chapter 10



# Reference Library Applications

---

### Chapter Overview

---

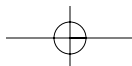
This chapter will focus on Reference Library applications. These applications are used to store information and tend to be used for reference purposes. This chapter includes two projects—a connection document manager and a spreadsheet report generator.

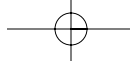
Reference Library applications can be used for a variety of purposes, such as to

- Store documents or files
- Store Web links
- Store lessons learned from projects
- Store labor hours
- Store postal codes
- Store names, addresses, and phone numbers
- Store and track company assets

By the end of the chapter you will have learned

- How to import and send connection documents
- How to send a connection document to a coworker
- How to send a form through email
- How to build a button to query a Domino server
- How to generate spreadsheets from a database view
- How to attach a file to an email
- How to remove a character from a string
- How to store a form in the document





---

## Reference Library Applications

---

In general, Reference Library applications can be quickly and easily built. This type of application typically stores static or historical information and tends to be used for reference purposes. In many cases, the application simply contains a form and view and can be built in a few hours.

This chapter includes two projects. The first project is designed to manage server connection documents. Connection documents are used by the Lotus Notes client and typically correct the error “Unable to find path to server.” This is a common problem encountered by users when multiple Domino servers are implemented.

The second project builds Microsoft Excel spreadsheets from any Lotus Notes view. This project supplements Reference Library applications. Here, users select a default view to be used to build the spreadsheet. The routine then calculates the number of columns, column names, and column widths to build a spreadsheet report. All columns that contain a field value are then added to the spreadsheet. When complete, this LotusScript library can be added to any new or existing Notes application.

### NOTE

You must have the Microsoft Office and Excel applications installed in order to generate spreadsheets from Lotus Notes database applications.

---

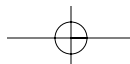
## Project A: Build a Connection Document Database

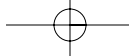
---

Difficulty Level:	Easy
Completion Time:	2 hours
Project Materials:	2 Forms, 1 View, 10 Action Buttons, 1 Role
Languages Used:	Formula Language, LotusScript
Possible Usages:	Management of server connection documents

### Architecture

This Reference Library application is designed to manage Lotus Notes connections to company-wide Domino servers. In many cases, companies utilize multiple Domino servers to host Lotus Notes applications. However, unless each server is located in the same Lotus Notes domain, users often receive the error “Unable to find path to server” when trying to open a database link.





Lotus Notes users typically receive this message because the client is unable to resolve, or figure out, the server address. In technical terms, the server address is referred to as either the Fully Qualified Domain Name (e.g., server01.raleigh.ibm.com) or the Internet Protocol Address (e.g., 9.1.67.100). In most cases, the creation of a server connection document will correct this error message and enable the user to connect to the server application.

Connection documents are stored in the user's Personal Address Book (PAB) in the Lotus Notes client. Here's how they work. When the user attempts to access a server, or an application on a server, the client checks the PAB for the server address. If the server information is found, then the Lotus Notes client uses this information to connect to the server. If the server information is not stored in the PAB, then a general query (also known as a network broadcast) is sent across the network to try to find the server address. If the server address cannot be determined, then the "Unable to find path to server" message is displayed.

This application offers one solution to manage Domino server connection documents. The purpose of this application is to provide a simple method to create, manage, and distribute IP addresses to the Lotus Notes client. Features of this application include

- A central repository of server connection information
- The ability for administrators to add and update server connections
- The ability for anyone to send a connection document to a coworker
- The ability for anyone to import or refresh existing server connections

Using this program, users can send or retrieve the most current connection documents and have them stored in their PAB. In most cases, this will ensure that users can access a server or a database on that server.

There are three primary features for this application—managing connection documents, sending a single connection document, and importing all connection documents. First, this application restricts the ability to create or edit connection document information. Only application administrators will have this authority. Second, anyone will have the authority to select a connection document and email it to a coworker. Finally, the application will include a feature to update or refresh all server connection documents that are stored in the user's PAB.

### Create the Database

To start this project, launch the Lotus Domino Designer client. When the Designer client is running, select the **File > Database > New** menu options. Specify an application title and file name (see Figure 10.1). Be sure to select **-Blank-** as the template type.

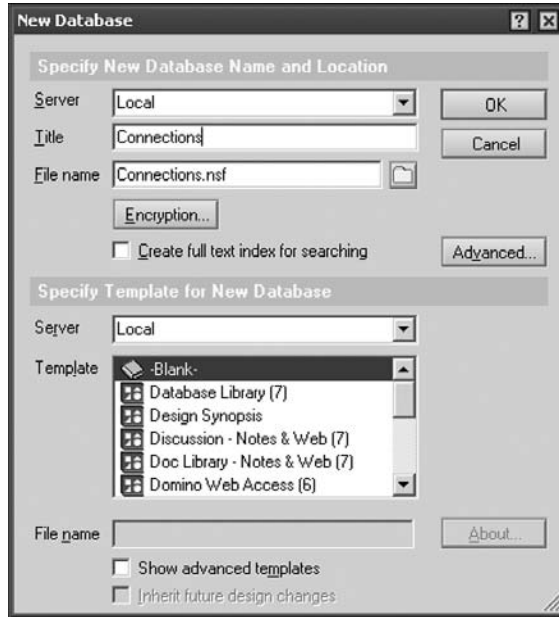


Figure 10.1 New Database dialog

### Set the Database Roles

After the database is created, start by establishing the roles associated with the database. Roles enable specific people to perform actions that others cannot. In this case, the “Admin” role will enable the application administrator to create new server connection documents. Anyone who does not have this role will not have this authority.

To create the role, select the **File > Database > Access Control** menu options. By default, the Basics tab should be active. Switch to the **Roles** tab and select the **Add** button (located at the bottom of the dialog window).

This will display a new popup window called Add Role (see Figure 10.2). Type **Admin** and click the **OK** button to add the role.

Click **OK** a second time to close the Access Control List (ACL) window.

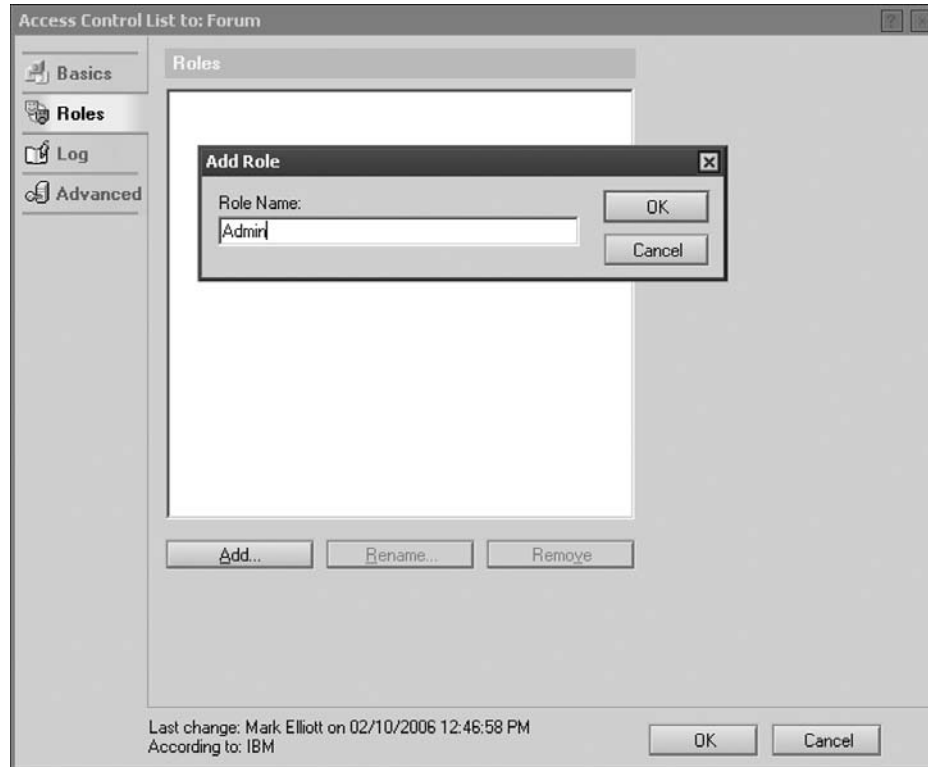
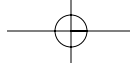


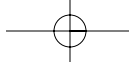
Figure 10.2 Add Role dialog

## Create the Connection Form

This form is used to store Domino server connection information and includes five fields—server name, Internet Protocol (or fully qualified domain) address, port, connect type, and updates. These fields will be used to build or update server connection documents in the user's PAB. All fields on the form are required. The general public will be able to open the documents. Only users assigned the "Admin" role will be permitted to create or update connection information in the database.

To create the form, select the **Create > Design > Form** menu options. Give the form a descriptive title at the top of the form—such as **Domino Server Connection**—and add the following text field descriptions down the left side of the form.

- Server Name:\*
- Server IP:\*
- Connect Type:\*
- Connect Port:\*
- Last Updated:



Next, create the following fields using the **Create > Field** menu options. Be sure to set the data type, formula, and other attributes for each field on the form using the properties dialog box and/or Programmer's pane.

Field Name	Type	Default Value	Formula	Remarks
ServerName	Text, Editable			
ServerIP	Text, Editable			
Type	Text, Editable	"Local Area Network"		
Port	DialogList, Editable	"TCP/IP"		On tab 2, select <b>Use formula for choices</b> and add the following in the dialog window formula field. @GetPortsList ( [Enabled] )
Updated	Date/Time, Computed		value := @If (@IsDocBeingEdited; @Now; @ThisValue); value	On tab 2, select the <b>Display Time</b> setting. Make sure both the date and time options are selected.

### QuerySave Event

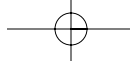
The `QuerySave` event will be used to verify that all required fields contain a valid value. In this case, the subroutine verifies that none of the fields are blank. Using the `Continue` built-in design flag, the document can only be saved if this variable contains a `True` value at the completion of the subroutine. If any field contains an invalid value—blank in this case—the `Continue` variable is set to `False`, and the document is not saved. Locate the `QuerySave` event found in the Objects pane for the form and insert the following code.

```
Sub Querysave(Source As NotesUIDocument, Continue As Variant)
```

```
    Dim doc As NotesDocument
    Dim MsgText As String
    Set doc = source.Document
    Continue = True
    MsgText = ""
```

```
    If Trim(doc.ServerName(0)) = "" Then
        MsgText = MsgText + "Specify a Server Name." + Chr$(13)
    End If
```

```
    If Trim(doc.ServerIP(0)) = "" Then
```



```
MsgText = MsgText + "Specify an IP Address." + Chr$(13)
End If

If Trim(Doc.Type(0)) = "" Then
    MsgText = MsgText + "Specify a Connect Type." + Chr$(13)
End If

If Trim(Doc.Port(0)) = "" Then
    MsgText = MsgText + "Specify a Connect Port." + Chr$(13)
End If

If MsgText <> "" Then
    MsgBox MsgText,16,"Required Fields."
    Continue = False
End If

End Sub
```

The next step will be to create the action buttons for the form—Edit, Save, Close, and Query Server. These buttons will manage the various document transactions within the application.

### Edit Button

The Edit button will change the document from read mode to edit mode. Select the **Create > Action > Action** menu options and name the button **Edit**. Now switch to tab 2 in the properties dialog and select both the **Previewed for editing** and **Opened for editing** checkboxes. Close the properties dialog and add the following to the Programmer's pane. After the code has been added, save and close the action.

```
@Command([EditDocument])
```

### Save Button

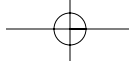
The Save button will cause the document to be saved to the database. This button will only display if the document is in edit mode. Select the **Create > Action > Action** menu options and name the button **Save**. Now switch to tab 2 in the properties dialog and select both the **Previewed for reading** and **Opened for reading** checkboxes. Close the properties dialog and add the following to the Programmer's pane. After the code has been added, save and close the action.

```
@Command([FileSave])
```

### Close Button

The Close button, as the name implies, closes the document and returns the user to the previously opened view. Select the **Create > Action > Action** menu options and name the button **Close**. Close the properties dialog and add the following to the Programmer's pane.

```
@Command([FileCloseWindow])
```



## Query Server Button

The Query Server button will issue the Ping command to query whether the server is connected to the network. The result of the query is displayed to the user in a text message.

Select the **Create > Action > Action** menu options. In the properties dialog, name the button **Query Server** and close the dialog window. Then, change the Language Selector from **Formula** to **LotusScript** and add the following in the Click event.

```
Sub Click(Source As Button)

    Dim w As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Dim doc As NotesDocument
    Dim statement As String
    Dim fileName As String
    Dim fileNum As Integer
    Dim result As String
    Dim text As String

    Set uidoc = w.CurrentDocument
    Set doc = uidoc.Document
    fileName$ = "c:\pingtext.txt"
    statement$ = "command.com /c ping " + _
        doc.ServerIP(0) + " > " + fileName$
    result = Shell ( statement, 6)
    Sleep( 10 )
    result = ""

    fileNum% = Freefile()
    Open fileName$ For Input As fileNum%
    Do While Not Eof( fileNum% )
        Line Input #fileNum%, text$
        result = result + text$ + Chr$(13 )
    Loop
    Close fileNum%
    MsgBox result , , "Result From Server Query"
    Kill fileName$

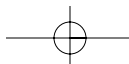
End Sub
```

The form should look like Figure 10.3 in the Lotus Domino Designer client.

To complete the form configuration, you will need to define who can create documents with this form. For this project, only those users assigned the “Admin” role will be able to create or modify documents. Select the **Design > Form Properties** menu options (see Figure 10.4).

Set the form name to **Connection|ConDoc** on tab 1 and then switch to tab 6. In the “**Who can create documents with this form**” section, uncheck the “**All authors and above**” option and select only the **[Admin]** option.

Save and close the form.



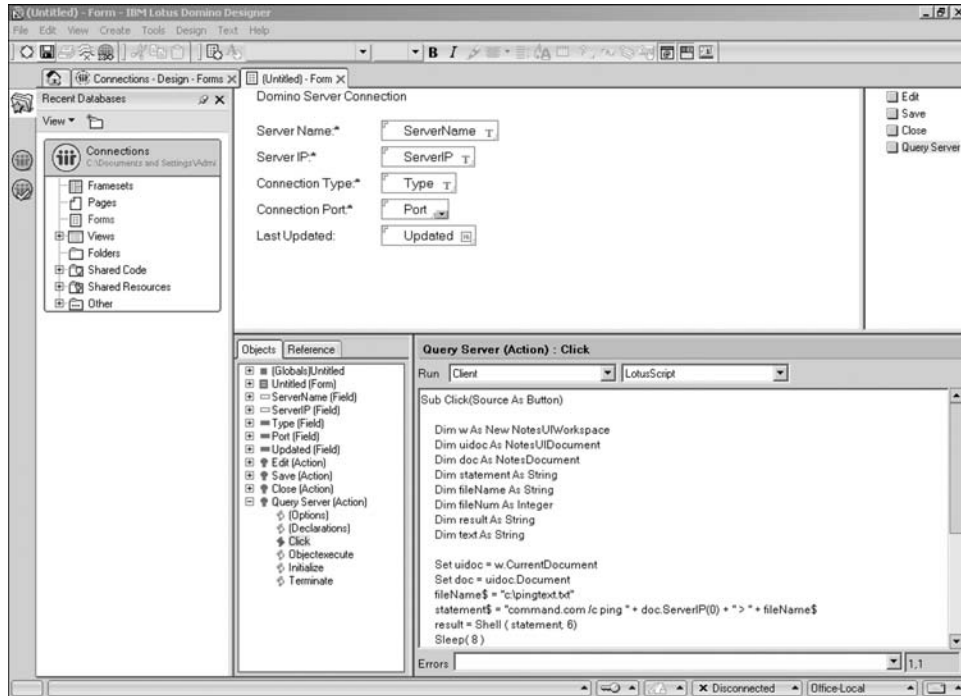
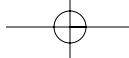
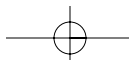
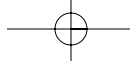


Figure 10.3 Connection Document form



Figure 10.4 Security tab of the Form properties dialog





## Create the Memo Form

The Memo form will be used to email a server connection document to another person. When the server connection document is received, the recipient can use the Import Connection button to add or update the server connection document in their PAB.

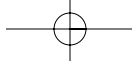
All design information associated with the form will be included in the email that's sent to the user. The user will have the option to import the connection or cancel the request. The design information must be sent in the email in order for the Import and Cancel buttons to display in the user's mail database.

To create the form, click the **New Form** button or select the **Create > Design > Form** menu options. After the form has been created, add the following field label descriptions down the left side of the form, leaving a couple blank lines after the **Subject** and the **Instructions to recipient** label.

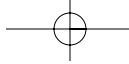
- Send To:
- Copy To:
- Subject:
- Instructions to recipient:

Next, create the following fields using the **Create > Field** menu options. Be sure to set the data type, formula, and other attributes for each field on the form using the properties dialog box and/or Programmer's pane.

Field Name	Type	Default Value Formula	Remarks
SendTo	Names, Editable	@UserName <i>By default, the email will be sent to the person sending the connection. If you do not want the "SendTo" field to have a default, remove this formula.</i>	Select <b>Allow multiple values</b> in tab 1 and <b>Use Address Dialog for choices</b> in tab 2.
CopyTo	Names, Editable		Select <b>Allow multiple values</b> in tab 1 and <b>Use Address Dialog for choices</b> in tab 2.
Subject	Text, Editable		
Body	Rich Text, Editable	"This note contains a Domino Server Connection Document. Select the 'Import Connection Doc'" + @NewLine +	



Field Name	Type	Default Value Formula	Remarks
		"button in the action bar to import the document into your personal address book. In most cases," + @NewLine + "importing the connection document will resolve network connectivity problems associated with the " + @NewLine + "specified Lotus Notes server. " + @NewLine + @NewLine	
Sign	Text, Computed	"1"	Signs the memo email with the user ID credentials of the person sending the connection document. Create this field at the very bottom of the form. Change the font color of the field to <b>RED</b> to indicate that this is a hidden field.
SaveOptions	Text, Computed	"0"	Prevents the memo from being saved in the database. Add this field next to the Sign field. In tab 6, check the <b>Hide paragraph if formula is true</b> checkbox and set the formula to 1. Change the font color of the field to <b>RED</b> to indicate that this is a hidden field.



## Create the Send Button

The Send button will transmit the server connection document to all people listed in the SendTo and CopyTo fields. This button will only be visible to the person sending the connection document and will not be visible to the email recipients. Select the **Create > Action > Action** menu options to create the button.

In the properties dialog, name the action button **Send**. Then switch to tab 2, check the **Hide paragraph if formula is true** checkbox, and set the formula to the following in the properties dialog. When complete, close the properties dialog.

```
!@IsNewDoc
```

Next, change the Language Selector from **Formula** to **LotusScript** and add the following in the Click event.

```
Sub Click(Source As Button)

    Dim w As New NotesUIWorkspace
    Dim uidoc As NotesUIDocument
    Set uidoc = w.CurrentDocument
    Call uidoc.Send
    Call uidoc.Close

End Sub
```

## Create the Cancel Button

The Cancel button will cancel the current transaction. This button closes the current memo and also cancels the connection document import for the user who will eventually receive the memo. Select the **Create > Action > Action** menu options. In the properties dialog, name the action button **Cancel** and close the properties dialog. Next, add the following formula in the Programmer's pane.

```
@Command([FileCloseWindow])
```

## Create the Import Connection Button

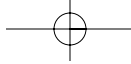
The Import Connection button updates the connection document information in the user's PAB. Select the **Create > Action > Action** menu options. In the properties dialog, name the action button **Import Connection**. Then switch to tab 2, check the **Hide paragraph if formula is true** checkbox, and set the formula to the following in the properties dialog. When complete, close the properties dialog.

```
@IsNewDoc
```

Next, change the Language Selector from **Formula** to **LotusScript** and add the following in the Click event.

```
Sub Click(Source As Button)

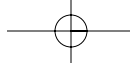
    Dim s As NotesSession
```



```
Dim w As New NotesUIWorkspace
Dim db As NotesDatabase
Dim view As NotesView
Dim uidoc As NotesUIDocument
Dim docA As NotesDocument
Dim docB As NotesDocument
Dim answer as Integer
Set uidoc = w.CurrentDocument
Set docB = uidoc.Document
Set s = New NotesSession
Set db = New NotesDatabase ("", "Names.nsf")
Set view = db.GetView ( "Connections" )
Set docA = view.GetDocumentByKey ( docB.ServerName(0), True)

'-----
' Create document if none exist
'-----
Dim docC As New NotesDocument( db )
If docA Is Nothing Then
    docC.Form = "local"
    docC.Destination = docB.ServerName(0)
    docC.LanPortName = docB.Port(0)
    docC.PortName = docB.Port(0)
    docC.OptionalNetworkAddress = docB.ServerIP(0)
    docC.PhoneNumber= docB.ServerIP(0)
    docC.Type = "Connection"
    docC.ConnectionType = "0"
    docC.Source = "*"
    docC.ConnectionLocation = "*"
    docC.Save True, True
    MsgBox "Connection document created." , , "Success"
Else
'-----
' Ask to replace document if one exists
'-----
    answer% = MsgBox ("An existing connection was found."+_
"Replace the existing document?", 36, "Continue?")
    If answer% = 6 Then
        docA.Remove ( True )
        docC.Form = "local"
        docC.Destination = docB.ServerName(0)
        docC.LanPortName = docB.Port(0)
        docC.PortName = docB.Port(0)
        docC.OptionalNetworkAddress = docB.ServerIP(0)
        docC.PhoneNumber= docB.ServerIP(0)
        docC.Type = "Connection"
        docC.ConnectionType = "0"
        docC.Source = "*"
        docC.ConnectionLocation = "*"
        docC.Save True, True
        MsgBox "Connection document created." , , "Success"
    End If
End If

End Sub
```



## Update Form Properties

In order for the Import Connection and Cancel buttons to display in the user's memo, the design information for this form must be included in the email notification. This is accomplished through the properties dialog for the form.

Select the **Design > Form Properties** menu options (see Figure 10.5). Name the form **Memo|Memo**. Next, uncheck the **Include in menu** and **Include in Search Builder** options and select the **Store form in document** option.

The screenshot shows the 'Form Properties' dialog box. The 'Name' field is 'Memo | Memo'. The 'Type' is 'Document'. Under 'Display', 'Include in menu', 'Include in Search Builder', and 'Include in Print' are unchecked. Under 'Versions', 'Versioning' is 'None' and 'Create versions' is 'None'. Under 'Options', 'Store form in document' is checked, while 'Default database form', 'Disable Field Exchange', 'Automatically refresh fields', 'Anonymous Form', 'No Initial Focus', 'No Focus On F6', 'Sign Documents that Use This Form', 'Do not add field names to field index', and 'Allow Autosave' are unchecked. 'Render pass through HTML in Notes' is checked. Under 'Conflict Handling', 'Create Conflicts' is selected.

**Figure 10.5** Form properties dialog

The form should look similar to Figure 10.6.

Save and close the form.

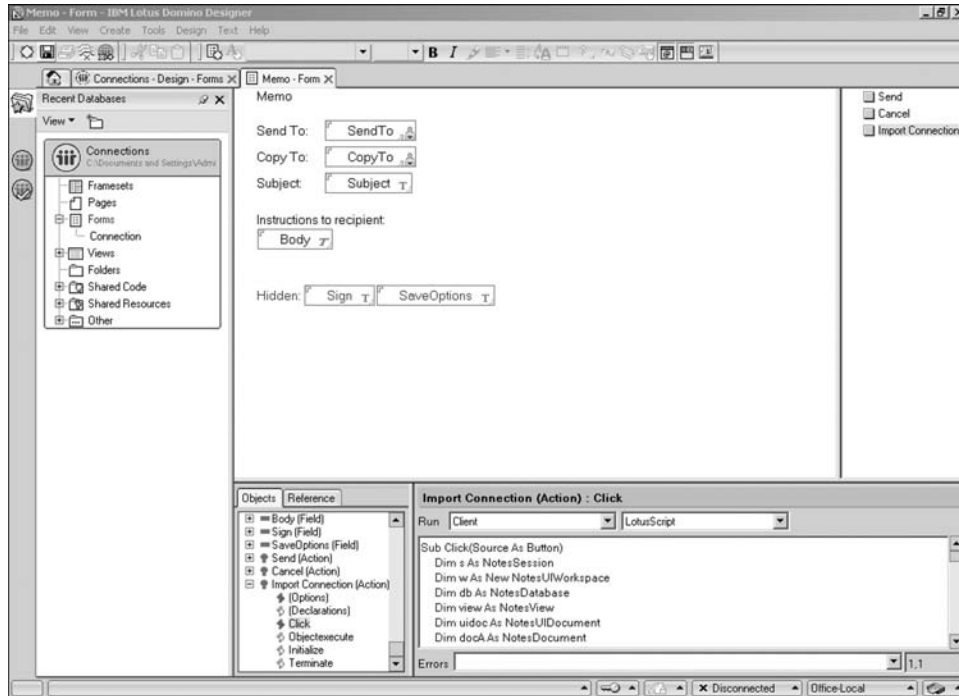
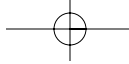


Figure 10.6 Memo form

## Create the Servers View

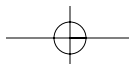
By default, a view called (*untitled*) is automatically created when the database is first created. To configure this view, navigate to Views in the Design pane and double-click on the view called “(untitled)”. When the view is displayed, the Designer client will immediately display the properties dialog for the view. Specify **Servers** as the view name and alias name. Close the properties dialog.

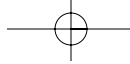
### Column 1

This column will list the server name for all documents stored in the application database. Double-click on the default column and rename the column title to **Server** in the properties dialog. Switch to tab 2 and set the sort order to **Ascending**.

To set the column value, change the display type from **Simple Function** to **Field** and select the following field.

ServerName





### Column 2

The second column will list the associated IP address or fully qualified domain name for the server. Select the **Create > Append New Column** menu options to add the column. In the properties dialog, set the column name to **Address** and column width to **20** on tab 1. Change the display type from **Simple Function** to **Field** and select the following field.

```
ServerIP
```

### Column 3

The last column displays the last modification date of the connection document. Select the **Create > Append New Column** menu options to add the column. In the properties dialog, set the column name to **Updated** and width to **15** on tab 1. Change the display type from **Simple Function** to **Field** and select the following field.

```
Updated
```

### Create the New Connection Button

The application administrator will use the New Connection button to create new server connection documents in the database. This button will not be available to the general public. Select the **Create > Action > Action** menu options. In the properties dialog, set the button name to **New Connection** in tab 1. Next, switch to tab 2 to select the **Hide action if formula is true** option and add the following to the formula window of the properties dialog.

```
!@IsMember(" [Admin] ";@UserRoles)
```

Close the properties dialog and add the following in the Programmer's pane.

```
@Command([Compose]; "ConDoc")
```

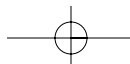
#### NOTE

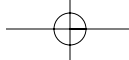
This button will not be visible when accessing the database in "local" mode. Administrators must access the application on the server in order for the button to appear. However, by selecting the **Enforce consistent ACL** option on the **Advanced** tab of the ACL properties, the users with the "Admin" role will be able to see the button.

### Create the Send Connection Button

This action button will allow users to send a single connection document to one or more coworkers. Select the **Create > Action > Action** menu options. In the properties dialog, set the button name to **Send Connection** in tab 1.

Next, change the Language Selector from **Formula** to **LotusScript** and add the following in the Programmer's pane in the **Click** event.





```

Sub Click(Source As Button)

    Dim s As NotesSession
    Dim w As New NotesUIWorkspace
    Dim db As NotesDatabase
    Dim dc As NotesDocumentCollection
    Dim uidoc As NotesUIDocument
    Dim docA As NotesDocument
    Dim docB As NotesDocument
    Dim text As String
    Set s = New NotesSession
    Set db = s.CurrentDatabase
    Set dc = db.UnprocessedDocuments
    Set docA = dc.GetFirstDocument

    If dc.Count <> 1 Then
        MsgBox "Please select only one connection document."
    Else
        Call w.ComposeDocument ( "", "", "Memo" )
        Set uidoc = w.CurrentDocument
        Set docB = uidoc.Document
        Call docA.CopyAllItems( docB, True )
        Call docB.RemoveItem("Authors")
        docB.SignOnSend = True
        Call uidoc.Reload
        text$ = " Server Connection for " + docA.ServerName(0)
        Call uidoc.FieldSetText( "Subject", text$ )
    End If

End Sub

```

## Create the Refresh Connections Button

This button allows users to add or update all server connection documents stored in their PAB. When clicked, the application will compare the documents stored in the database with that of the address book. If the document does not exist, a new connection document will be created in the PAB. If the document already exists, the information will be replaced.

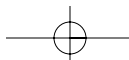
Select the **Create > Action > Action** menu options. In the properties dialog, set the button name to **Refresh Connections** in tab 1. Next, change the Language Selector from **Formula** to **LotusScript** and add the following in the Programmer's pane in the Click event.

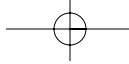
```

Sub Click(Source As Button)

    Dim s As NotesSession
    Dim w As New NotesUIWorkspace
    Dim db As NotesDatabase
    Dim view As NotesView
    Dim viewA As NotesView
    Dim viewB As NotesView
    Dim docA As NotesDocument
    Dim docB As NotesDocument
    Dim dc As NotesDocumentCollection

```





```

'-----
' Get list of connections in application
'-----
Set s = New NotesSession
Set db = s.CurrentDatabase
Set viewA = db.GetView( "Servers" )
Set docA = viewA.GetFirstDocument

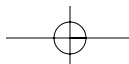
'-----
' Locate the connections view in the PAB
'-----
Set db = New Notesdatabase ("", "Names.nsf")
Set view = db.GetView ( "Connections" )

'-----
' Loop through the list of connections
'-----
While Not ( docA Is Nothing )
    Print "Checking: " + docA.ServerName(0)
    Set docB = view.GetDocumentByKey(docA.ServerName(0), True)
    If docB Is Nothing Then
        Dim docC As New NotesDocument( db )
        docC.Form = "local"
        docC.Destination = docA.ServerName(0)
        docC.LanPortName = docA.Port(0)
        docC.PortName = docA.Port(0)
        docC.OptionalNetworkAddress = docA.ServerIP(0)
        docC.PhoneNumber= docA.ServerIP(0)
        docC.Type = "Connection"
        docC.ConnectionType = "0"
        docC.Source = "*"
        docC.ConnectionLocation = "*"
        docC.Save True, True
        Print "Creating connection: " + docA.ServerName(0)
    Else
        If docA.ServerIP(0) <> docB.OptionalNetworkAddress(0) Then
            docB.Form = "local"
            docB.LanPortName = docA.Port(0)
            docB.PortName = docA.Port(0)
            docB.OptionalNetworkAddress = docA.ServerIP(0)
            docB.PhoneNumber= docA.ServerIP(0)
            docB.Type = "Connection"
            docB.ConnectionType = "0"
            docB.Source = "*"
            docB.ConnectionLocation = "*"
            docB.Save True, True
            Print "Updating connection: " + docA.ServerName(0)
        End If
    End If
    Set docA = viewA.GetNextDocument( docA )
Wend
Print "Updates complete."

End Sub

```

The view should look similar to Figure 10.7.



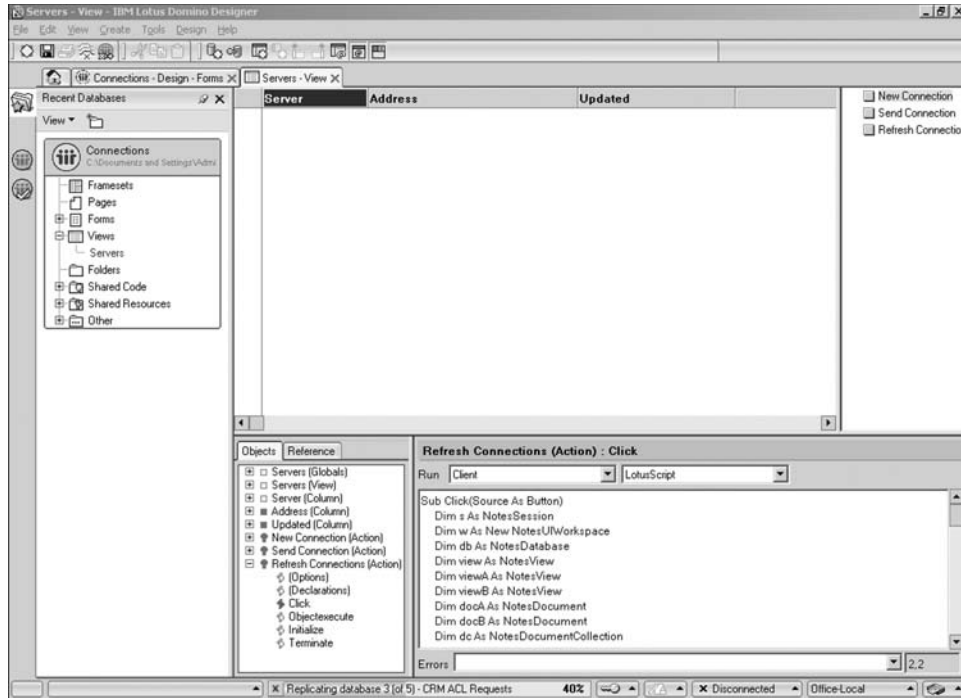
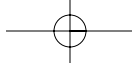


Figure 10.7 Server view

Save and close the view design element.

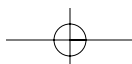
### Application Security

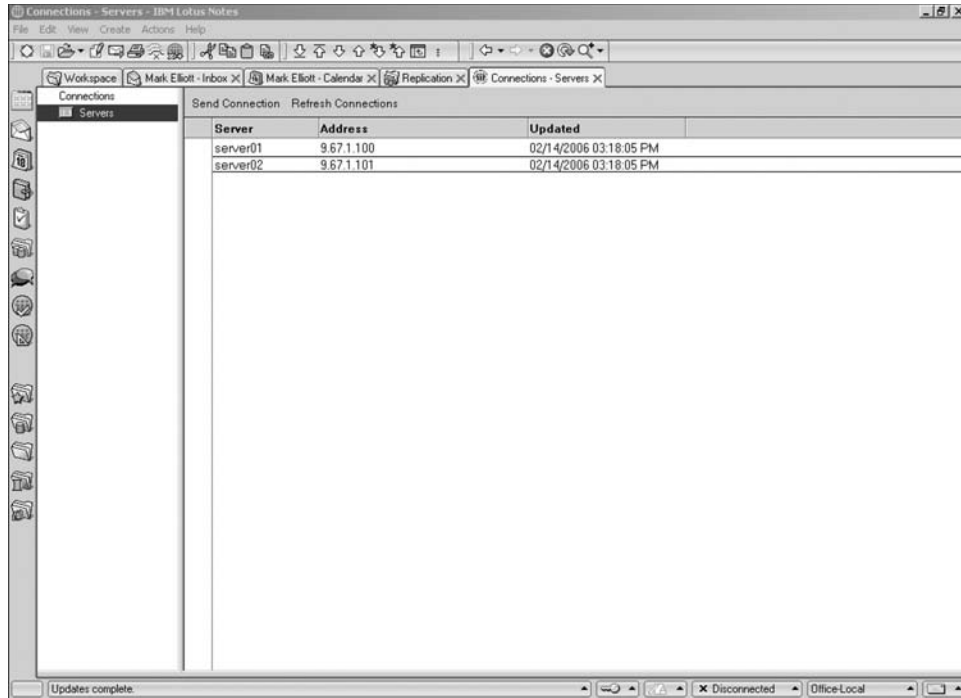
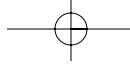
Application security is managed through the ACL and “roles.” Roles are used to give specific people the ability to view information or perform actions that others cannot. For this project, only users assigned the “Admin” role will have the authority to create and modify server connection documents.

To set the ACL settings, select the **File > Database > Access Control** menu options. By default, the Basics tab should be active. Click on the **-Default-** user in the center of the screen. Change the access level (right side) to **Author**.

Next, identify the person who will manage the application. Click the **Add** button (located at the bottom of the ACL dialog window) to add a user. After the user has been added, give the person **Editor** access and select the **[Admin]** role. Click **OK** to save the ACL settings (see Figure 10.8).

Congratulations, you’ve completed the project!





**Figure 10.8** Completed project

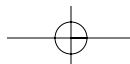
## Project B: Build a Spreadsheet Generator

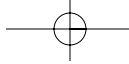
Difficulty Level:	Easy
Completion Time:	1 hour
Project Materials:	1 Form, 1 View, 1 Shared Action Button, 1 LotusScript Library
Languages Used:	LotusScript
Possible Usages:	Add to any Lotus Notes database to create reports

### Architecture

This project contains two Domino design elements—a shared action button and a LotusScript library. When built, these two design elements can be added to any Lotus Notes database to generate Microsoft Excel spreadsheets. Although this project is not a “Reference Library” application, you may want to include it in a reference or any database.

The first design element is a shared action button. This button triggers the `PromptUser` subroutine and asks the user to select a view to be used as the basis for the spreadsheet. Users have the option to select a view from the dropdown list or to cancel the transaction.





The second design element, the LotusScript library, contains all subroutines and functions required to produce a spreadsheet. This library consists of four subroutines and one function. The combination of these design elements performs the following tasks.

1. Scan the database and build a list of views
2. Prompt the user to select the view to be used as the basis for the spreadsheet
3. Scan the selected view to retrieve the view name, column titles, field values, and column widths
4. Use the view parameters to build the spreadsheet
5. Generate a unique file name based on the date and time stamp
6. Create the spreadsheet object
7. Format the cells of the spreadsheet by setting the font type and style
8. Parse the documents displayed in the view and add them to the spreadsheet
9. Close the spreadsheet
10. Email the spreadsheet to the user

**NOTE**

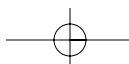
This script library is designed to work with columns that are set to a field value. All columns that contain a formula or simple action are ignored. It's important to note that the spreadsheet could include columns containing a formula and could also build graphs. However, this requires more advanced programming to generate. The point of this exercise is to generate a simple spreadsheet and to illustrate how the code can be applied to a database.

**NOTE**

This script library requires the Microsoft Excel product to be installed on the user's workstation. The user will receive an error if Microsoft Excel is not installed.

## Create the Database

To start this project, launch the Lotus Domino Designer client and create a blank database. When the Designer client is running, select the **File > Database > New** menu options (see Figure 10.9). Specify an application title and file name. Be sure to select **-Blank-** as the template type.



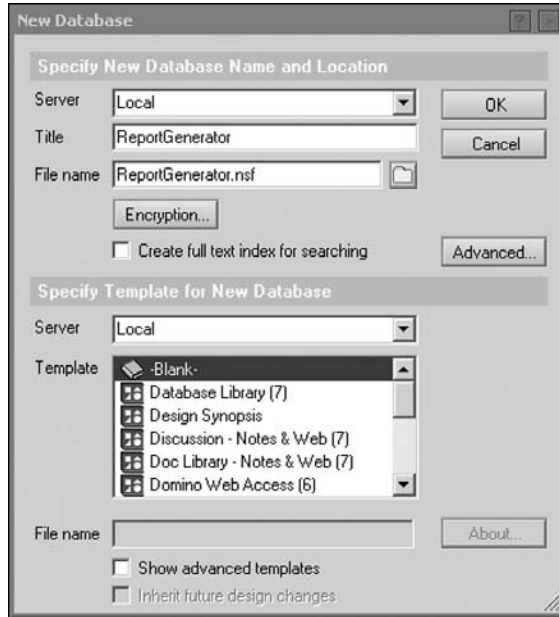
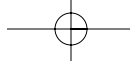


Figure 10.9 New Database dialog

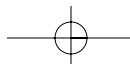
## Create the Spreadsheet Script Library

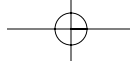
The LotusScript library will hold a number of common subroutines used to generate a Microsoft Excel spreadsheet based on a view in the Lotus Notes database.

### TIP

The following code is available in the developer's toolbox. Simply open the "Project Library" database in your Lotus Notes client and navigate to the "Script Library" section for the appropriate project. After you locate the project, copy and paste the code into the Designer client for the current project. Be sure the programmer's pane has focus before pasting the code. Alternatively, you can elect to manually type the code. You can then save and close the library. When prompted for a name, specify **ReportLibrary**.

To create the library, select the **Create > Design > Script Library > LotusScript Library** menu options.





## Declarations

Locate the “(Declarations)” section and add the following global statements in the Programmer’s pane. These objects will be used throughout the library subroutines and functions.

```
Dim s As NotesSession
Dim db As NotesDatabase
Dim view As NotesView
```

## Initialize Subroutine

The following statements are used to initialize the session and database objects. Add these statements in the Initialize section of the Programmer’s pane.

```
Sub Initialize

    Set s = New NotesSession
    Set db = s.CurrentDatabase

End Sub
```

## PromptUser Subroutine

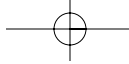
The PromptUser subroutine displays a list of views in the Notes application database to the user. The user then selects a view to be used as the basis for the spreadsheet. Type the following in the Programmer’s pane.

```
Sub PromptUser

    Dim w As New NotesUiWorkspace
    Dim view As NotesView
    Dim x As Integer
    Dim result As String

    '-----
    ' Build an array of views in the database
    '-----
    x=0
    If Not Isempty (db.Views) Then
        Forall v In db.Views
            Redim Preserve myList(x) As String
            myList(x) = v.Name
            x = x + 1
        End Forall
    End If

    '-----
    ' Ask the user to select a view to generate the spreadsheet
    '-----
    result$ = w.Prompt( PROMPT_OKCANCELCOMBO, +_
    "Make your choice","Select a view to build the spreadsheet",+_
    "Select View", myList)
    If result$ <> "" Then
        Print "Found view: " + result$
    End If
End Sub
```



```

        Call GenerateReport ( result$ )
    End If
    Print "Complete"

End Sub

```

### ***GenerateReport Subroutine***

This subroutine will parse the design elements in the selected view into arrays. This subroutine builds an array of column titles, column widths, and field values. It also counts the total number of columns and identifies the view name. These arrays and data values are then passed to another subroutine—*CreateSpreadsheet*—to build the spreadsheet file. Insert the following in the Programmer's pane.

```

Sub GenerateReport ( result As String )

    '-----
    '---- Build data arrays for the selected view
    '-----
    Dim x As Integer
    Set view = db.GetView( result$ )
    x=0
    Forall c In view.Columns
        If c.IsField Then

            ' Build an array of column names
            Redim Preserve ColumnName(x) As String
            ColumnName(x) = c.Title

            ' Build an array of column Widths
            Redim Preserve ColumnWidth(x) As Variant
            ColumnWidth(x) = c.width

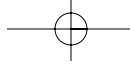
            ' Build an array of column field names
            Redim Preserve FieldName(x) As String
            FieldName(x) = c.ItemName

            x = x + 1
        End If
    End Forall

    '-----
    '---- Create spreadsheet based on the selected view.
    '---- Parm1 - array of all valid fields in the view
    '---- Parm2 - array of the column titles
    '---- Parm3 - array of the column widths
    '---- Parm4 - the name of the view
    '---- Parm5 - total number of columns in the view
    '-----
    Call CreateSpreadsheet ( FieldName , ColumnName, _
        ColumnWidth, view.Name, x-1 )

End Sub

```



### CreateSpreadsheet Subroutine

The CreateSpreadsheet subroutine is used to build the Microsoft Excel spreadsheet. This subroutine requires five parameters—an array of field names, an array of column titles, an array of column widths, a view name, and the total number columns in the view. These parameters define the content and layout of the spreadsheet. Type the following in the Programmer's pane.

```
Sub CreateSpreadsheet ( field As Variant, column As Variant, ColWidth As Variant,
Sheetname As String, cntr As Integer)
```

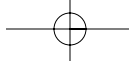
```

    On Error Goto oops
    Dim file As Variant
    Dim wksSheet As Variant
    Dim filename As String
    Dim alphabet(25) As String
    Dim cell As String
    Dim value As String
    Dim doc As NotesDocument
    Dim row As Long
    Dim x As Integer
    Dim n As Integer

    '-----
    '---- Build the filename for the spreadsheet
    '-----
    Dim theDate As String
    Dim theTime As String
    theDate = removeString ( Format(Date$, "Medium Date"), "-")
    theTime = removeString ( Format(Time$, "Long Time"), ":")
    theTime = removeString ( theTime, " ")
    Filename = "C:\Report" + "_" + theDate + "_" + theTime
    Print "Building file: " + Filename

    '-----
    '---- Build an array of the alphabet
    '-----
    For n = 65 To 90
        'Print "Letter" + Cstr(n-65) + " = " + Chr$(n)
        alphabet(n-65) = Chr$(n)
    Next

    '-----
    '---- Create the spreadsheet file object
    '-----
    Set file = CreateObject("Excel.Application")
    file.Visible = False
    file.DisplayAlerts = False
    file.Workbooks.Add
    Set wksSheet = file.Worksheets.Add
    wksSheet.name = Sheetname
    file.Worksheets("Sheet1").Delete
    file.Worksheets("Sheet2").Delete
    file.Worksheets("Sheet3").Delete
```



```
Set wksSheet = file.Worksheets( Sheetname )
wksSheet.Select

'-----
'---- Set the column width for first 26 columns
'-----
For x=0 To cntr
wksSheet.columns(Alphabet(x)).Columnwidth=Cint(colWidth(x)+5)
Next
Print "Set the default column width complete."

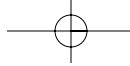
'-----
'---- Set font style for spreadsheet
'-----
With file.Range("A:Z")
    .WrapText = True
    .Font.Name = "Arial"
    .Font.FontStyle = "Regular"
    .Font.Size = 8
End With
Print "Set the font style complete."

'-----
'---- Set font style for header row
'-----
With file.Range("A1:Z1")
    .WrapText = True
    .Font.Name = "Arial"
    .Font.FontStyle = "Bold"
    .Font.Size = 8
End With
Print "Spreadsheet initialized."

'-----
'---- Load the spreadsheet with data
'-----
Print "Starting data load into spreadsheet. Please be patient"
Set view = db.GetView( SheetName )
Set doc = view.GetFirstDocument
row = 1

' Create the column title row
Set wksSheet = file.Worksheets( Sheetname )
wksSheet.Select
For x=0 To cntr
    cell$ = Alphabet(x) + Cstr( row )
    file.Range( cell$ ).Select
    file.Activecell.FormulaR1C1 = Column(x)
Next
row = 2

' Create the data rows
While Not(doc Is Nothing)
    Set wksSheet = file.Worksheets( Sheetname )
    wksSheet.Select
```



```
'Loop through each column and add data to the row
For x=0 To cntr
    cell$ = Alphabet(x) + Cstr( row )
    file.Range( cell$ ).Select
    file.Activecell.FormulaR1C1 =doc.GetItemValue(field(x))
Next
Set doc = view.GetNextDocument( doc )
row = row + 1
Wend
Print "Data load complete."

'-----
'---- Save, close and email file to the person
'-----
file.activeworkbook.saveas Filename
file.activeworkbook.close

' Comment out the following line to skip sending the email
SendReport ( Filename )
Print "Report sent."

' Comment out the following line to save file to the harddrive
Kill Filename+".xls"

Set file = Nothing
Exit Sub

oops:
Msgbox "Error" & Str(Err) & ": " & Error$
file.activeworkbook.close
Kill Filename+".xls"

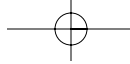
End Sub
```

**NOTE**

By default, this subroutine will email the spreadsheet to the user. However, if you prefer that the files be saved to the user's hard drive, comment out the following three statements: `SendReport ( Filename )`, `Print "Report sent."`, and `Kill Filename+".xls"`. This will cause the files to be saved to the user's hard drive in the C:\ directory.

**RemoveString Function**

This function removes all instances of a specific character from the target object string. This function checks each character in the string. If the current character does not match the search string, then the character is added to a temporary variable. If the character does match, then the character is skipped. The result is a rebuilt string that is returned to the calling subroutine. Insert the following in the Programmer's pane.



```
Function RemoveString ( object As String, SearchString As String) As Variant

    Dim tempString As String
    Dim j as Integer
    tempString = ""
    For j% = 1 To Len( object )
        If Mid$(object, j%, 1) <> SearchString Then
            tempString = tempString + Mid$(object, j%, 1)
        End If
    Next
    RemoveString = tempString

End Function
```

### ***SendReport* Subroutine**

The `SendReport` subroutine is used to create an email, attach the spreadsheet file, and send it to the person who generated the report. Type the following in the Programmer's pane.

```
Sub SendReport ( filename As String )

    Dim PersonName As New NotesName(s.UserName)
    Dim rtitem As NotesRichTextItem
    Dim object As NotesEmbeddedObject
    Dim doc As NotesDocument
    Set doc = New NotesDocument( db )
    doc.Form = "Memo"
    doc.SendTo = PersonName.Abbreviated
    doc.Subject = "Report - " + filename
    Set rtitem = New NotesRichTextItem( doc, "Body" )
    Call rtitem.AddNewline(1)
    Call rtitem.AppendText("Attached below is the requested report. ")
    Call rtitem.AddNewline(2)
    Set object = rtitem.EmbedObject ( EMBED_ATTACHMENT, "", Filename+".xls")
    doc.Send False
    MsgBox "The requested report has been sent to you.", 0, "Success"

End Sub
```

Save and close the LotusScript library. When prompted, name the library **ReportLibrary**.

### **Create the Generate Report Shared Action**

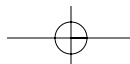
The shared action button prompts the user to select the view to be used to build the spreadsheet. To create the button, select the **Create > Design > Shared Action** menu options and name the button **Generate Report**. Close the properties dialog.

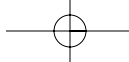
Next, change the Language Selector from **Formula** to **LotusScript**. Locate the **(Options)** section and insert the following in the Programmer's pane.

```
Use "ReportLibrary"
```

Add the following in the Click event.

```
Sub Click(Source As Button)
```





```
Call PromptUser
End Sub
```

Save and close the button when complete.

## Create the Contact Form

This form is being created for illustration purposes to demonstrate the `GenerateReport` subroutine. However, this form is not required to actually implement the spreadsheet script library. This form includes four fields—name, address, phone, and email.

To create the form, select the **Create > Design > Form** menu options. Give the form a descriptive title at the top of the form—such as **Contact**—and add the following text field descriptions down the left side of the form.

- Name:
- Address:
- Phone:
- Email:

Next, create the following fields using the **Create > Field** menu options. Be sure to set the data type, formula, and other attributes for each field on the form using the properties dialog box and/or Programmer's pane.

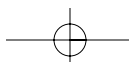
Field Name	Type	Default Value	Formula	Remarks
Name	Text, Editable			
Address	Text, Editable			
Phone	Text, Editable			
Email	Text, Editable			

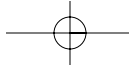
Select the **File > Save** menu options to save the file. When prompted, name the form **Contact | Contact**. Close the form after the file has been saved.

## Create the Contact View

This view will be used as the basis to demonstrate the `GenerateReport` functionality and is included for illustration purposes.

By default, a view called (*untitled*) is automatically created when the database is first created. To configure this view, navigate to Views in the Design pane and double-click on the view called "(untitled)". When the view is displayed, the Designer client will immediately display the properties dialog for the view. Specify **Contacts** as the view name and alias in tab 1. Close the properties dialog.





Next, click on the header for the predefined column and delete the column. To build the view, select the **Create > Append New Column** menu options to add four columns to the view. For each column, switch the Language Selector to **Field**. Set column one through four to the following field values:

- Name
- Address
- Phone
- Email

After the column value is specified, select **Design > Column Properties**. Click on the column header and specify a title in the properties dialog for each column.

Finally, select the **Create > Action > Insert Shared Action** menu options and insert the Generate Report shared action button. Save and close the view.

Congratulations! You have completed the project.

Add a couple contact documents and give the button a try. Note: You must have Microsoft Excel installed on your workstation in order for the report generator to work.

