Integrating IBM Lotus Notes with Microsoft Office using LotusScript and automation

Mike Maroney

June 29, 2009
(First published March 10, 2009)

Learn new techniques that you can apply to Microsoft® Office applications, such as Microsoft Excel, to publish information in an IBM® Lotus Notes® database in a variety of formats.

Editor's note: Know a lot about this topic? Want to share your expertise? Participate in the IBM Lotus software wiki program today.

IBM Lotus Notes wiki

Introduction

Publishing needs

Most Lotus Notes developers are familiar with the limitations of publishing in Lotus Notes. You might frequently need to create mail merges with data in Lotus Notes applications, to create reports with specific formatting, or to enable features such as pivot tables. These examples are not natively available in Lotus Notes.

An often overlooked option is to use Microsoft Office to publish information stored in Lotus Notes databases. Although the concepts presented in this article can be applied to sharing data with other applications (for example, Lotus SmartSuite®), the focus of this document is publishing with Microsoft Word.

Requirements and limitations

To use the code listed in this document, you must have the associated application (Microsoft Word, in these examples) installed. The samples are written as user-triggered agents that run on the local system.

The two methods presented in this document for creating references to other applications are not supported using OS/2, UNIX®, Linux®, and Macintosh.

Finally, use of OLE-automation in a server-side environment is not recommended nor supported by Microsoft. Instead, you can consider publishing your content using the open source XML file formats. Refer to Microsoft Support Article 257757 for more details.
Assumptions
This article assumes that you are familiar with LotusScript® and the concept of object models.

To view the object model of a Microsoft Office application, follow these steps:

1. Open the application and press ALT + F11. This step opens the Visual Basic Editor.
2. Select View - Object Browser. This step allows you to review the complete list of objects and their associated properties and methods. This article includes snippets of code examples and provides an example database that you can download.

Getting started
Creating references to Microsoft Office applications

The first step in sharing data between Lotus Notes and other applications is to get a reference to the target application. LotusScript has two functions available that provide this capability:

CreateObject( className )
GetObject( pathName, [ className ] )

You can find more details, including examples, about each of these functions in the Lotus Domino® Designer Help database.


CreateObject creates a reference to the OLE automation object supplied in the className parameter. Here is a list of some of the Microsoft Office class names:

- Word.Application
- Excel.Application
- Access.Application
- PowerPoint.Application
- MSProject.Application

Each application has a general purpose and can be used to publish information in different ways. Microsoft Excel, for example, can be used to publish sales data and create charts. Microsoft Project can be used to display tasks on a timeline.

As stated earlier, this article focuses on using Microsoft Word to publish data. To launch and create a programmatic handle to the Microsoft Word application, use the following code:

Dim wordApp as Variant
Set wordApp = CreateObject(“Word.Application”)  
wordApp.Visible = True

GetObject provides a reference to an application that is already running. The pathName parameter can be an empty string, or it can reference a file. If you reference a file, the className is not required because it is determined by the type of file.
In both functions, you obtain a reference to the application that you specify. At this point, you can now begin controlling that application through LotusScript, performing such actions as creating or opening documents, passing text, and formatting text.

The last line of the previous code snippet makes the application visible. This step is generally done after your code has finished processing, but you might find it helpful during development and for debugging purposes. If the application is visible, you can step through each line of code as it runs.

Creating a new Microsoft Word document

Continuing from the previous code snippet, you can pass text to Microsoft Word using LotusScript. First, however, you need to get a reference to a new document in Word. Use this code:

```livescript
Dim wordDoc as Variant
Set wordDoc = wordApp.Documents.Add()
```

In the second line, you navigate down the Microsoft Word object model from the application object to the collection of documents in that application. From there, use the add method, which creates a new document and returns a reference to it.

The best source for available objects and their associated methods and properties is the target application. For example, start Microsoft Word and then select Tools - Macro - Visual Basic Editor from the menu (or you can press Alt + F11). After the editor is open, select View - Object browser from the menu (or you can press F2). Select the application object, which is the top-level object, and on the right side are the associated members as shown in figure 1.

**Figure 1. Object browser**

![Object browser](image)

Using the select method

The document object contains many of the methods that you might want to review. Before you begin entering text in the document, however, you need to think about where the cursor is. Even though you are performing actions on the document programmatically, Microsoft Word behaves as if a person were manually interacting with it.
For example, when you create a new document in Microsoft Word, the cursor displays at the top of the page. The same thing occurs when you create a document programmatically. Therefore, let’s take a look at the Select method:

```vba
Dim cursor as Variant
Set cursor = wordApp.Selection
```

These two lines of code provide a reference to the Selection object. Next, you can use the Selection object to type text into the document.

**Typing text**

The Selection object includes the TypeText method, which is used to pass a string of characters to the document at the current cursor location.

```vba
cursor.TypeText("Hello, world!")
```

This code passes the ubiquitous "Hello, world!" text string to the active document at the current cursor location.

**Formatting**

When you want to format text in Microsoft Word, you select the text and then apply your formatting changes. For example, you might want to apply a bold typeface to the "Hello, world!" string that you just entered.

```vba
wordDoc.Paragraphs(1).Range.Select
cursor.Font.Bold = True
```

The first line selects the first paragraph in your document. The second line applies bold formatting to the selection.

**Using bookmarks**

Microsoft Word includes a feature called bookmark, which offer users the opportunity to create labels for a position in a document. This feature allows the user to jump quickly to that location by selecting Edit - Go to. As a developer, you can also employ this feature to streamline document navigation and references.

In the code shown in listing 1, you create three sections, create a bookmark for each, and then use those bookmarks to identify where you want to insert text.

**Listing 1. Creating and using bookmarks**

```vba
For counter = 1 To 3
    selection.typetext "This is Section Header " & Cstr(counter)
    selection.typeparagraph
    selection.typeparagraph
    selection.moveup
    selection.moveup
    selection.expand wdParagraph
```
Set bookmarks(counter) = worddoc.bookmarks.add("SectionHeader" & Cstr(counter), selection.range)
selection.moveend wdStory
selection.collapse wdCollapseEnd
Next

selection.goto wdGoToBookmark, , , "SectionHeader1"
selection.movedown
selection.typetext "This text is associated with the first section...."
selection.goto wdGoToBookmark, , , "SectionHeader2"
selection.movedown
selection.typetext "This text is associated with the second section...."
selection.goto wdGoToBookmark, , , "SectionHeader3"
selection.movedown
selection.typetext "This text is associated with the third section...."

In summary, you follow a process beginning with the highest level of the object model and work your way down. Using Variant variables you get references to different objects that you frequently use. Review the complete code shown in listing 2. You can copy and paste this code into an agent and step through it with the LotusScript debugger to see it in action.

**Listing 2. HelloWorld sample code**

' Dimension the variables
Dim wordApp as Variant
Dim wordDoc as Variant
Dim cursor as Variant

' Get references to OLE objects
On Error Resume Next
Set wordApp = GetObject("Word.Application")
If wordApp Is Nothing Then
 Set wordApp = CreateObject("Word.Application")
End If
wordApp.Visible = True
Set wordDoc = wordApp.Documents.Add()
Set cursor = wordApp.Selection

' Pass text to the new document and format it
cursor.TypeText("Hello, world!")
wordDoc.Paragraphs(1).Range.Select
cursor.Font.Bold = True

**Heavy examples**

The next three sections discuss using more advanced features of Microsoft Word, including tables, dialog boxes, and mail merges. You can use tables to organize your information; they are also used in mail merges. Dialog boxes let you obtain input from the user. For example, you can ask the user to indicate where a file should be saved.

**Creating and using tables**

Tables in Microsoft Word offer you the ability to align and organize your data. In the case of a mail merge, you define a data source document, which typically is a document with a table. Each row in the table represents a record, while each column in the table represents a field.

The example in listing 3 shows you how to add a new table with a single row and seven columns. Each column is populated with a field name.
## Listing 3. Inserting a table

```vba
Set tbl = worddoc.tables.add(selection.range, 1, 7)
'Set up the header row of the table
With tbl
    .cell(1, 1).select
    selection.typetext "Title"
    .cell(1, 2).select
    selection.typetext "FirstName"
    .cell(1, 3).select
    selection.typetext "LastName"
    .cell(1, 4).select
    selection.typetext "OfficeStreetAddress"
    .cell(1, 5).select
    selection.typetext "OfficeCity"
    .cell(1, 6).select
    selection.typetext "OfficeState"
    .cell(1, 7).select
    selection.typetext "OfficeZip"
    .rows.last.select
End With
```

You can also format the table for legibility. Microsoft Word uses points for measurement. You might find it easier to create a function to perform the translation of inches to points by multiplying the number of inches by 72 (there are 72 points in one inch). The code shown in listing 4 uses just such a function to adjust the column widths. This example also makes use of a constant called wdAdjustNone, which has a value of 0. Although this is an inherent constant in Microsoft Word, you need to declare your own constants.

## Listing 4. Adjusting column width

```vba
'Format column widths
With tbl
    .columns(1).setwidth InchesToPoints(.6), wdAdjustNone
    .columns(2).setwidth InchesToPoints(1), wdAdjustNone
    .columns(3).setwidth InchesToPoints(1), wdAdjustNone
    .columns(4).setwidth InchesToPoints(1.6), wdAdjustNone
    .columns(5).setwidth InchesToPoints(1), wdAdjustNone
    .columns(6).setwidth InchesToPoints(1), wdAdjustNone
    .columns(7).setwidth InchesToPoints(1), wdAdjustNone
End With
```

## Using common dialog boxes

Microsoft Word includes easy access to common dialog boxes, such as Save As. You can utilize them by calling the Word.Application.Dialogs.Show method and identifying which one you want to display. For example, the code snippet here displays the Save As dialog box, which allows users to indicate where and under what file name they want to save the document.

```vba
rc = wordobj.dialogs(wdDialogFileSaveAs).Show
```

In this example, rc is set to the return value of the dialogs().Show call. Table 1 outlines the possible values returned.

## Table 1. Possible File Save As dialog box return values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>The Close button</td>
</tr>
</tbody>
</table>
Creating a mail merge

Microsoft Word offers excellent mail merge capabilities. A mail merge is generated by creating a form letter and associating with it a data source document. The source document contains multiple records, normally in a table, and fields that are inserted into the form letter. After it is defined, you can merge the data into a new document or send it directly to the printer.

The example shown in listing 5 can be used to select Person documents in a Lotus Notes Personal Address book and then to export those to a Microsoft Word data source document used by the form letter. **NOTE:** Some variable declarations have been removed due to space limitations. Full source code is accessible by downloading the associated sample file in the Downloads section of this article.

### Listing 5. Creating a mail merge

```vbscript
Option Public
Option Explicit
%include "LCONST.LSS"

Dim w As notesuiworkspace
Dim uiview As notesuiview
Dim col As notesdocumentcollection, doc As notesdocument
Dim tbl As Variant
Dim maindoc As Variant
Dim path As String
Dim rc As Integer
Dim wordobj As Variant
Dim worddoc As Variant
Dim selection As Variant, range As Variant

Sub Initialize
  Set w = New notesuiworkspace
  Set uiview = w.currentview
  Set col = uiview.documents
  If col.count = 0 Then
    MsgBox "No documents selected!", MB_OK + MB_ICONINFORMATION, "Error"
  End If
  Set wordobj = createobject("Word.Application")
  wordobj.visible = True
  Set worddoc = wordobj.documents.add()
  Call worddoc.content.select()
  Set selection = wordobj.selection()
  Call selection.collapse()
  path = wordobj.Options.DefaultFilePath(wdDocumentsPath)
  Call CheckForExistingFiles
  Call FormatMailMergeSource

  Set doc = col.getfirstdocument
  Do Until doc Is Nothing
```

-1 The OK button
0 (zero) The Cancel button
> 0 (zero) A command button: 1 is the first button, 2 is the second button, and so on
Call PopulateData
Set doc = col.getnextdocument(doc)
Loop
Msgbox "Your data source has been created. You now need to identify where you wish to save this file.", _
MB_OK + MB_ICONINFORMATION, "Save the Data Source"
rc = wordobj.dialogs(wdDialogFileSaveAs).Show
If (rc = 0) Or (rc = -2) Then
  MsgBox "You have cancelled your mail merge.", _
  MB_OK + MB_ICONINFORMATION, "Operation Cancelled"
End If
End If
Call CreateMailMergeMain
End Sub

Sub FormatMailMergeSource()
'Page layout
With worddoc.pagesetup
  .topmargin = inchestopoints(.4)
  .bottommargin = inchestopoints(.4)
  .leftmargin = inchestopoints(.4)
  .rightmargin = inchestopoints(.4)
End With
Set tbl = worddoc.tables.add(selection.range, 1, 7)
'Create the header row of the table
With tbl
  .cell(1, 1).select
  .cell(1, 2).select
  .cell(1, 3).select
  .cell(1, 4).select
  .cell(1, 5).select
  .cell(1, 6).select
  .cell(1, 7).select
End With
'Format column widths
With tbl
  .columns(1).setwidth InchesToPoints(.6), wdAdjustNone
  .columns(2).setwidth InchesToPoints(1), wdAdjustNone
  .columns(3).setwidth InchesToPoints(1), wdAdjustNone
  .columns(4).setwidth InchesToPoints(1.6), wdAdjustNone
  .columns(5).setwidth InchesToPoints(1), wdAdjustNone
  .columns(6).setwidth InchesToPoints(1), wdAdjustNone
  .columns(7).setwidth InchesToPoints(1), wdAdjustNone
End With
selection.collapse wdCollapseStart
End Sub

Sub PopulateData()
    tbl.rows.last.select
    selection.insertrowsbelow 1
    With tbl.rows.last
        .cells(1).range.insertafter doc.Title(0)
        .cells(2).range.insertafter doc.FirstName(0)
        .cells(3).range.insertafter doc.LastName(0)
        .cells(4).range.insertafter doc.OfficeStreetAddress(0)
        .cells(5).range.insertafter doc.OfficeCity(0)
    End With
End Sub
This listing builds on some of the previous sections to generate a mail merge. You could use similar code to create a mail merge to send customers information about new products, current account balances, or invitations to participate in special programs.

**Conclusion**

These techniques can be applied to other Microsoft Office applications, such as Microsoft Excel. As long as you have the software installed, you can use similar techniques to publish information in a Lotus Notes database in a variety of formats. Remember that the Object Explorer in each application provides valuable documentation and examples about the object model.

**Acknowledgment**

The author would like to thank Josh Sherman, also in the Lotus Technical Support organization, for his technical review of this article.
Appendix: Description of sample application

The sample Lotus Notes application, included in the Download section of this article, includes the following agents:

- **Basic example.** This agent is the ubiquitous "Hello, World!" example, which sends the text string to a new Microsoft Word document.
- **Timing tricks part A.** This agent and the next one work together to demonstrate the potential pitfalls of timing. The Part A agent enters some text, formats it, then inserts a hard return. It shows how the formatting gets carried down to the next paragraph.
- **Timing tricks part B.** This agent and the previous one work together to demonstrate the potential pitfalls of timing. The Part B agent enters some text, inserts a hard return, enters additional text, then moves up and formats the original text.
- **Bookmarks example.** This agent creates three sections, bookmarks them, and later uses those bookmarks to insert text.
- **Mail merge.** This agent allows users to select one or more Lotus Notes documents in the database's People view, creates a Microsoft Word data source document, creates a main document linked to the data source document, and then generates a mail merge in Microsoft Word. It sets up the data source and main documents; the user then needs to type the form letter and to insert the fields using normal mail merge procedures in Microsoft Word.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSOffice.nsf</td>
<td>1MB</td>
</tr>
</tbody>
</table>
Related topic

- Refer to the IBM Lotus Notes and Domino product page on developerWorks® Lotus.

© Copyright IBM Corporation 2009
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)