

Hummingbird Enterprise[™] 2004

5.1.0.5

COM Automation for Microsoft Word & Excel Version: 5.1.0.5 Service Release 5 and later Copyright © 1998-2007 Hummingbird Ltd. All rights reserved. Electronic Publication Date: January 2007

Hummingbird Enterprise 2004, DM, RM, SearchServer, DM WorkFlow, Imaging, KM, Enterprise Webtop, Collaboration, Web Publishing, DOCSFusion, and DOCS Open are trademarks of Hummingbird Ltd. and/or its subsidiaries. All other copyrights, trademarks, and trade names are the property of their respective owners.

Your enclosed license agreement with Hummingbird Ltd. or one of its affiliates specifies the permitted and prohibited uses of the product. Any unauthorized duplication or use of the product in whole or part is strictly forbidden. No part of this document may be copied, reproduced, translated, or transmitted in any form or by any means without the prior written consent of Hummingbird Ltd.

RESTRICTED RIGHTS LEGEND. Unpublished rights reserved under the copyright laws of the United States and any other appropriate countries. The SOFTWARE is provided with restricted rights. Use, duplications, or disclosure by the U.S. Government is subject to restriction as set forth in subparagraph (c) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights clause at 48 CFR 52.227-19, as applicable, similar clauses in the FAR and NASA FAR Supplement, any successor or similar regulation.

The information contained in this document is subject to change without notice. If this document is provided in both printed and electronic form, the electronic form will govern in the event of any inconsistency. Information in this document is subject to change without notice and does not represent a commitment on the part of Hummingbird Ltd. Not all copyrights pertain to all products.

DISCLAIMER. Hummingbird Ltd. software and documentation have been tested and reviewed. Nevertheless, Hummingbird Ltd. makes no warranty or representation, either express or implied, with respect to the software and documentation included. In no event will Hummingbird Ltd. be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the software or documentation included with these products. In particular, Hummingbird Ltd. shall have no liability for any programs or data used with these products, including the cost of recovering such programs or data.

Corporate Headquarters 1 Sparks Avenue, Toronto, Ontario M2H 2W1 Canada U.S./Canada Toll-free: 1 877 FLY HUMM (359 4866) Tel: 1 416 496 2200, Fax: 1 416 496 2207, Website: www.hummingbird.com

Contents

Chapter 1	Using COM Automation for Microsoft Word	
	About COM Automation for Microsoft Word 5	
	Supporting Software 6	
	New Functionality 6	
	Installing COM Automation for Microsoft Word 7	
	Create a Launch Method 7	
	Refresh DM Server Caches 10	
	Implement COM Automation for Word on User Workstations 1	0
	Performing a Mail Merge in Word 12	
	Before You Begin 12	
	Selecting the Main Document 12	
	Creating a New Data Source in Word 13	
	Selecting a Saved Data Source 14	
	Opening a Native Data Source 15	
	Customizing the DM Footer with COM Automation 16	
	Microsoft Word Customized Footer Sample Code 18	
	Using Word's AutoRecover Feature 24	
	Troubleshooting COM Automation for Word 25	
	Word Stops Responding 25	
	Auto-Recovery 25	
	Previous Integration Files Must Be Removed 25	
	COM Automation for Word Registry Setting 25	
	E-mail Integration 25	
Chapter 2	Using COM Automation for Microsoft Excel	
	About COM Automation for Microsoft Excel 27	
	Supporting Software 28	
	New Functionality 28	
	Installing COM Automation for Microsoft Excel 29	
	Create a Launch Method 29	

	Refresh DM Server Caches 31	
	Implement COM Automation for Excel on User Workstations	32
	Customizing the DM Footer with COM Automation 34	
	Microsoft Excel Customized Footer Sample Code 35	
	Using Excel's Auto-Recovery Feature 40	
	Troubleshooting COM Automation for Excel 41	
	Previous Integration Files Must Be Removed 41	
Chapter 3	Using Microsoft Office Menu Shortcut Keys with COM Automation	
	Microsoft Office Intercepted Shortcut Keys 44	
	Word Intercepted Shortcut Keys 44	
	Excel Intercepted Shortcut Keys 45	
	Removing DM Menu Items 46	
	DM Intercepted Menu Items 46	
	Active Integration 46	
	Passive Integration 47	
Chapter 4	DM Simplified API Technical Information	
	DM Simplified API 49	
	Technical Specifications 50	
	Functions 50	
	DM.OpenDoc 50	
	DM.CloseDoc 51	
	DM.SaveDoc 51	
	DM.SaveDocAs 52	
	DM.NewDoc 53	
	DM.IsDMDoc 53	
	DM.GetDocInfo 53	
	DM.Library 54	
	Sample Code 55	
	DM.OpenDoc 55	
	DM.CloseDoc 55	
	DM.SaveDoc 57	

DM.SaveAs **58** DM.IsDMDoc **59** DM.GetDocInfo **60**

Chapter 5 DM COM Add-in Programmer's Guide for Microsoft Word DM_COM_Addin.clsWord.DMClose 62 DM COM Addin.clsWord.DMCompare 62 DM_COM_Addin.clsWord.DMCompareAndMerge 62 DM_COM_Addin.clsWord.DMExit 63 DM COM Addin.clsWord.DMFooter 63 DM COM Addin.clsWord.DMInsertFile 63 DM_COM_Addin.clsWord.DMInsertPicture 64 DM_COM_Addin.clsWord.DMNativeSave 64 DM COM Addin.clsWord.DMNativeSaveAs 65 DM_COM_Addin.clsWord.DMOpen 65 DM COM Addin.clsWord.DMPasswordProtect Document 65 DM_COM_Addin.clsWord.DMPrint 66 DM COM Addin.clsWord.DMSave 66 DM COM Addin.clsWord. DMSaveAS 66 Sample Code **68** DM COM Addin.clsWord.DMClose 68 DM_COM_Addin.clsWord.DMCompare 68 DM_COM_Addin.clsWord.DMCompareAndMerge 69 DM COM Addin.clsWord.DMExit 69 DM_COM_Addin.clsWord.DMFooter 70 DM COM Addin.clsWord.DMInsertFile 70 DM COM Addin.clsWord.DMInsert Picture 71 DM COM Addin.clsWord.DMNative Save 71 DM COM Addin.clsWord.DMNative SaveAs 72 DM_COM_Addin.clsWord.DMOpen 72 DM_COM_Addin.clsWord.DMPasswordProtectDocument 73 DM COM Addin.clsWord.DMPrint 73

DM_COM_Addin.clsWord.DMSave **74** DM_COM_Addin.clsWord.DMSaveAS **74**

Chapter 6 DM COM Add-in Programmer's Guide for Microsoft Excel DM_COM_ADDIN.clsExcel.DMClose 76 DM_COM_ADDIN.clsExcel.DMCloseWorkspace **76** DM_COM_ADDIN.clsExcel.DMExit 76 DM_COM_ADDIN.clsExcel.DMFooter 77 DM_COM_ADDIN.clsExcel.DMInsertPicture 77 DM_COM_ADDIN.clsExcel.DMNativeSave 77 DM COM ADDIN.clsExcel.DMNativeSaveAs 78 DM_COM_ADDIN.clsExcel.DMOpen 78 DM COM ADDIN.clsExcel.DMOpenVersion 79 DM_COM_ADDIN.clsExcel.DMPwdProtDoc 79 DM_COM_ADDIN.clsExcel.DMRestoreWorkspace **79** DM_COM_ADDIN.clsExcel.DMSave 79 DM_COM_ADDIN.clsExcel.DMSaveAS 80 DM_COM_ADDIN.clsExcel.DMSaveWorkspace 81 Sample Code 82 DM_COM_ADDIN. clsExcel.DMClose 82 DM_COM_ADDIN. clsExcel.DMClose Workspace 82 DM_COM_ADDIN.clsExcel.DMExit 83 DM COM ADDIN. clsExcel.DMFooter 83 DM_COM_ADDIN. clsExcel.DMInsert Picture 83 DM_COM_ADDIN. clsExcel.DMNativeSave 83 DM COM ADDIN. clsExcel.DMNativeSaveAs 84 DM_COM_ADDIN. clsExcel.DMOpen 85 DM_COM_ADDIN. clsExcel.DMOpen Version 85 DM_COM_ADDIN. clsExcel.DMPwd ProtDoc 85 DM_COM_ADDIN. clsExcel.DM RestoreWorkspace **85** DM_COM_ADDIN. clsExcel.DMSave 86 DM_COM_ADDIN. clsExcel.DMSaveAS 86 DM_COM_ADDIN. clsExcel.DMSave Workspace 87

Using COM Automation for Microsoft Word

About COM Automation for Microsoft Word

COM Automation for Microsoft Word uses a Microsoft Office COM Add-in to provide active and passive integration with DM. This add-in is meant to remove the dependency on Microsoft's built-in ODMA implementation. The add-in communicates with DM through DMmanaged COM interfaces and has no reliance on ODMA.

If you are using any customized items for Word (templates, VBA macros, Word add-ins), third-party add-ons, or other applications that depend on the Word ODMA implementation, then you should contact Technical Support before using Word Active COM Add-in integration.

This chapter contains the following sections:

- Installing COM Automation for Microsoft Word
- Performing a Mail Merge in Word
- Customizing the DM Footer with COM Automation
- Using Word's AutoRecover Feature
- Troubleshooting COM Automation for Word

Supporting Software

COM Automation for Microsoft Word is supported with DM 5.1.0.5 Service Release 5 and later.

This feature is supported for the following versions of Microsoft Word:

- Word 2000 SP3
- Word XP SP3
- Word 2003 SP2

New Functionality

- Performance enhancements, especially in startup.
- Ability to use Word as an e-mail editor.
- The DM mail merge function has been rewritten. Refer to Performing a Mail Merge in Word for instructions.
- A new Password Confirmation dialog box prompts you to enter the document's password when you select **File>Save** With Options.
- A new **Insert>Footer** menu option has been added for Active integration. A new **DM>Footer** menu item has been added for Passive integration.

Refer to the Customizing the DM Footer with COM Automation section for more information.

• Word's AutoRecover feature is integrated with DM. See Using Word's AutoRecover Feature for more information.

Installing COM Automation for Microsoft Word

Create a Launch Method

You can create a launch method through DM Webtop or by using the Library Maintenance tool, which is accessible on the DM server computer or any computer where the Server Admin Tools are installed.

Creating a Launch Method in DM Webtop

Following are instructions on how to create a launch method in DM Webtop.

To create a launch method in DM Webtop:

- 1. Click DM Admin>Validation Table>Applications.
- 2. Click **MS WORD** to display the settings for Microsoft Word.
- 3. Click Launch Methods.
- 4. Click **Add**. (Alternatively, you can click **Copy** and modify an existing launch method.)
- 5. In the **Description** field, enter Mi crosoft Word COM Automation (Active/Passive Integration).
- 6. In the **Location** field, enter the path where the **winword.exe** executable is located.
- 7. Enter the following launch method settings:

Command Line Parameters /ND

Integration Type

Full Integration

Leave the **Command Line Parameters** field blank to make Microsoft Word open with a blank document. The /ND setting, which indicates "no document," is typically used when Front End Profiling is enabled.

- 8. Select the **Enabled** check box.
- 9. Select **DDE Settings**. You will be prompted to save the changes to the launch method.
- 10. Enter the DDE Settings as follows:

Open Settings:

Application Name

WINWORD

Topic Name	SYSTEM
Command	[DDE.DDEOpen("%1")]
Command (not running)	[DDE.DDEOpen("%1")]

Print Settings:

Application Name	WINWORD
Topic Name	SYSTEM
Command	[DDE.DDEPrint ("%1", "1")]
Command (not running)	[DDE.DDEPrint ("%1", "0")]

- 11. Click **Save** and **Close** to complete the launch method for COM Automation for Microsoft Word.
- 12. Select **Save** again to save the application settings for Microsoft Word.

Creating a Launch Method in Library Maintenance

Following are instructions on how to create the launch method in Library Maintenance.

To create a launch method in Library Maintenance:

- 1. Click Start>Programs>Hummingbird>DM Server>Server Admin Tools>Library Maintenance.
- 2. Click Applications.
- 3. Double-click MS WORD.
- 4. In the Application Maintenance window, click Launch Methods.
- 5. In the Application Launch Methods window, click **Copy**.
- 6. Enter the following information:

Enabled: Ensure that this check box is selected.

Description: Enter a description for the launch method, such as COM Automation for Microsoft Word (Active/Passive).

Location: Ensure that the location to the **winword.exe** executable file is specified.

Command Line Parameters: Leave the **Command Line Parameters** field blank to make Microsoft Word open with a blank document, or enter /ND, which indicates to open with "no document."

Default Directory: Leave this field blank.

Search Mapping: Leave this field blank.

Integration: Select Full Integration.

- 7. Click **Apply**; the **DDE Settings** button becomes available.
- 8. Click **DDE Settings**; ensure that the DDE settings match the settings specified below:

Open Settings:

Application Name	WINWORD
Topic Name	SYSTEM
Command	[DDE.DDEOpen("%1")]
Command (not running)	[DDE.DDEOpen("%1")]

Print Settings:

Application Name	WINWORD
Topic Name	SYSTEM
Command	[DDE.DDEPrint ("%1", "1")]
Command (not running)	[DDE.DDEPrint ("%1", "0")]

- 9. Click Close.
- 10. Click **OK** in the Application Launch Methods window.
- 11. Click **Close**.
- 12. Click the **Options** button from the Application Maintenance dialog box.
- 13. Ensure that the **Shows on Desktop** and **Valid on Profile** check boxes are selected; click **OK**.
- 14. Click **OK**; the launch method for COM Automation for Word is now complete.

Refresh DM Server Caches

After you create the launch method for COM Automation for Microsoft Word, you must refresh the DM Server caches. On the DM Server machine, go to the Server Manager's **Caches** tab and select **Refresh All**.

Implement COM Automation for Word on User Workstations

You can install the COM Automation Add-in for Word from DM Webtop or by using the DM Extensions Server Setup. Instructions follow for both methods.

Using DM Extensions Server Setup

To implement COM Automation for Microsoft Word in DM Extensions Server Setup:

- 1. Click **Start>Programs>Hummingbird>DM Extensions Server Setup>Install DM Extensions**; the Installation Wizard appears and the installation begins.
- 2. Accept the default location specified to install the media, or click **Browse** to enter an alternation location; click **Next**.

If you have already installed DM Extensions, select the **Modify** option, click **Next**, and skip to step 4 below.

- 3. In the Choose Destination Location window, accept the default Destination Folder in which to install DM Extensions, or click the **Browse** button to specify an alternate location; click **Next**.
- 4. The Select Optional Components window appears; click the **DM Application Integration** plus (+) sign to expand the tree.
- 5. Expand the Microsoft Applications tree.
- 6. Expand the **Microsoft Word** tree.
- 7. Expand the Microsoft Word Active Integration tree.
- 8. Expand the via COM Automation tree.
- 9. Select one of the following check boxes:
 - with Front-End Profiling
 - with Front-End Profiling and Cost Recovery
 - Active Integration without Front-End Profiling

If you want to install COM Automation using Passive integration, in step 7, expand the **Microsoft Word Passive Integration** tree, and then click the **via COM Automation** check box.

- 10. Click Next.
- 11. In the Connection Information window, enter then machine name of the DM Server, or click **Browse** to locate the DM Server; then click **Next**.
- 12. The Start Copying Files window appears; click **Next** to start the installation.
- 13. When the installation is complete, click **Finish**.

Using DM Webtop

To implement COM Automation for Microsoft Word in DM Webtop:

- 1. In DM Webtop, click **My Options**.
- 2. Click the **Optional Components** tab.
- 3. Click the Install icon; the Installation Wizard appears.
- 4. Accept the default location specified to install the media, or click **Browse** to enter an alternation location; click **Next**.

If you have already installed DM Extensions, select the **Modify** option, click **Next**, and skip to step 6 below.

- In the Choose Destination Location window, accept the default Destination Folder in which to install DM Extensions, or click the Browse button to specify an alternate location; click Next.
- 6. The Select Optional Components window appears; click the **DM Application Integration** plus (+) sign to expand the tree.
- 7. Expand the Microsoft Applications tree.
- 8. Expand the Microsoft Word tree.
- 9. Expand the Microsoft Word Active Integration tree.
- 10. Expand the **via COM Automation** tree.
- 11. Select one of the following check boxes:
 - with Front-End Profiling
 - with Front-End Profiling and Cost Recovery
 - Active Integration without Front-End Profiling

If you want to install COM Automation using Passive integration, in step 9, expand the **Microsoft Word Passive Integration** tree, and then click the **via COM Automation** check box.

12. Click Next.

- 13. If the Connection Information window appears, enter then machine name of the DM Server, or click **Browse** to locate the DM Server; then click **Next**.
- 14. The Start Copying Files window appears; click **Next** to start the installation.
- 15. When the installation is complete, click **Finish**.

Performing a Mail Merge in Word

When using either Active or Passive Integration for Microsoft Word, use the steps below to perform a mail merge operation. You must enable a default launch method to use this feature; otherwise, the mail merge process will not work correctly.

Before You Begin If you will not be using an existing document for a data source, you should create a data source before beginning the mail merge operation. Compatible data sources include outside data sources such as Excel worksheets, delineated text files, or Outlook Contacts export files. Refer to Microsoft Word help for a full list of compatible data sources.

You can create a data source in Word or use a saved data source. Word data sources are created during the merge process. Refer to Opening a Native Data Source for more information.

Selecting the Main
DocumentThe first step in starting the mail merge is to select a Main Document.To select a Main Document:
1. Start Microsoft Word.

- 2. Select **View>Toolbars>Mail Merge** to display the Mail Merge toolbar, if not already visible.
- 3. Select the appropriate menu command depending on your version of Word:

Word Version	Active Integration	Passive Integration
Word 2000	Tools>DM Mail Merge	DM>Mail Merge
Word XP and 2003	Tools>Letters and Mailings> DM Mail Merge	DM>Mail Merge

The Hummingbird DM Mail Merge dialog box appears.

4. If you want to open a saved document, click **Select Main Document**. (If you are not already logged on to DM, you will be prompted to do so.) The Quick Retrieve dialog box appears, in which you can select a valid mail merge Main Document.

Select a valid mail-merge Main Document and click **OK**.

- 5. If you want to use the document that is currently open, click **Use Current Document**; the active document is set as the Main Document.
- 6. The next step is to select a data source containing the information that will populate the data fields in your document. Continue with one of the following procedures:
 - * Creating a New Data Source in Word
 - ✤ Selecting a Saved Data Source
 - * Opening a Native Data Source

To create a new data source:

- 1. Perform the steps in the Selecting the Main Document section.
- 2. In the Mail Merge Helper, click **Select Data Source;** a Quick Retrieve dialog box appears.
- 3. Click **Cancel** to display the Microsoft Word **Mail Merge Helper** dialog box.
- 4. Under Data Source, click Get Data>Create Data Source.
- 5. In the Create Data Source dialog box, specify the fields to be included in the data source.
- 6. When you have completed selecting the fields, click **OK**.
- 7. Save the data source to DM or locally.
- 8. After the document is saved, a Word message appears, stating that the data source is blank; select **Edit Data Source**.
- 9. The data source document will appear; enter data in all of the fields.
- 10. When you have completed entering data, save the document.

Creating a New Data Source in Word

Selecting a Saved Data Source

- In the Main Document, click Insert Merge Field on the Mail Merge toolbar anywhere that you want to insert a field. If necessary, click Insert Word Field to place fields such as <<Next Record>>, etc.
- 12. Complete any formatting in the Main Document.
- 13. Click Merge Helper.
- 14. On the Mail Merge toolbar, click one of the following options: Merge to a New document, Merge to Printer, or Merge to E-mail.

You will need to select which records to be used to complete the merge. When merging to an e-mail, you will need to select the field to be used as the e-mail address, enter a subject and select the e-mail format.

- 15. Click **OK**; the merged output is created.
- If you selected to merge to a new document, you can select File>Save As to profile the document in DM.

To select a saved data source:

- 1. Perform the steps in the Selecting the Main Document section.
- 2. In the Mail Merge Helper, click **Select Data Source**; a Quick Retrieve dialog box appears.
- 3. Select the previously saved data source file and click **OK**.

You can select another Word document, Excel workbook or other profiled document that is a compatible data source for a Microsoft Word mail merge.

- 4. Select the correct document version and click **OK**.
- 5. Depending on the type of data source selected, you might be prompted to confirm the type of data source. Select the data source type and click **OK**.
- 6. For Excel spreadsheets, you might be prompted to indicate which part of your spreadsheet you want to select. Select an item (or group of items) and click **OK**.
- 8. Complete any formatting in the Main Document.

	Press All+Tab to open the Hummingbird DM Mail Merge dialog box.	
	10. Click Merge Helper; the Microsoft Word Mail Merge Helper dialog box appears with the Main Document and Data Source fields populated with the documents you selected.	
	If you want to edit your Main Document and/or Data Source, click Edit next to the desired document. If you edit either document and choose to save your changes, the documents are uploaded to DM; however, this is valid for Word data sources only.	
	When you edit either document, in order to display the Merge Helper dialog box and continue with the merge, click Merge Helper on the Hummingbird DM Mail Merge dialog box.	
	11. Click Merge; the Merge dialog box appears.	
	12. Under Merge to, select one of the following options: New Document, Printer or Electronic Mail. Alternately, on the Mail Merge toolbar, select the Merge to a New document, a Merge to Printer, or Merge to E-mail toolbar button.	
	You will need to select which records to be used to complete the merge. When merging to an e-mail, you will need to select the field to be used as the e-mail address, enter a subject and select the mail format.	
	13. Click OK ; the merged output is created. If you selected the New Document option, you can select File>Save As to profile the document in DM.	
Opening a Native Data Source	Native data sources can include any compatible data source file that have not been profiled and saved in a DM library. These can include Excel workbooks, Access databases files, delimited text files and many more.	
	To open a native data source:	
	1. Perform the steps in the Selecting the Main Document section.	
	2. In the Mail Merge Helper, click Select Data Source; a Quick Retrieve dialog box appears.	
	3. Click Cancel to display the Mail Merge Helper dialog box.	

- 4. Under **Step 1, Main Document**, select **Create** and choose the document type you will be creating (Form Letters, Mailing Labels, etc.).
- 5. Under the Data source drop-down list, click Get Data Source>Open Data Source.
- 6. A native Open Data Source dialog box appears; select the document type from the **Files of type** drop-down box.

Locate and select a natively-saved data source document that contains data fields you want to insert in the main document and click **Open**.

- 7. Depending on the type of data source selected, you might be prompted to confirm the type of data source. Select the data source type and click **OK**.
- 8. For Excel spreadsheets, you might be prompted to indicate which part of your spreadsheet you want to select; select an item (or group of items) and click **OK**.
- 9. Select Edit the Main Document.
- 11. Complete any formatting in the main document.
- 12. On the Mail Merge toolbar, click one of the following options: Merge to a New document, Merge to Printer, or Merge to E-mail.

You will need to select which records to be used to complete the merge. When merging to an e-mail, you will need to select the field to be used as the e-mail address, enter a subject and select the mail format.

13. Click **OK**; the merged output is created. If you selected the **New Document** option, you can select **File>Save As** to profile the document in DM.

Customizing the DM Footer with COM Automation

You might have customized Word footers based on DM's macro functions, which are built upon ODMA. Because COM Automation does not use ODMA, these particular customizations will no longer work; however, you can customize Word footers using COM Automation. You should be familiar with Visual Basic for Applications (VBA) and the Microsoft VBA editor before performing the following steps.

To create a custom Word footer using COM Automation:

- 1. Open Microsoft Word.
- 2. If a blank document does not appear, create one by selecting **File>New**.
- 3. Save the document as a Document Template (*.dot) to the local file system. The name should not conflict with any other macro file names, such as **normal.dot**. Do not save the document in any of the Word **Startup** directories, as this will prevent the document being edited using the Microsoft VBA editor.
- 4. Open the Microsoft VBA editor by pressing ALT+F11 or by selecting Tools>Macros>Visual Basic Editor.
- 5. In the VBA Project Explorer, select the project containing the current document.
- 6. Right-click the **Modules** node below this project and select **Insert>Module**.
- 7. In the **Properties** window, rename the module. Ensure that the file name does not conflict with any other file names.
- 8. Add a **Public** subroutine; the name of the subroutine can be anything that VBA allows and does not conflict with other names.
- 9. In this subroutine, add your custom code that will insert the desired information into the footer. The process of inserting the footer information involves calls to the Microsoft Word object model. Depending on what you want inserted into the footer, the subroutine might also contain calls to the DM API, the DM Extensions API, other third-party APIs, and VBA functions. Refer to Microsoft Word Customized Footer Sample Code for more information.
- 10. From the VBA editor menu, select File>Save.
- 11. Select File>Close; focus returns to Word.
- 12. From the Word menu, select **File>Exit**.
- 13. Copy the .dot file to one of the Word Startup directories.

	The per-user Word Startup directory is located in C:\Documents and Settings\<user_name>\Application</user_name> Data\Microsoft\Word\STARTUP . You must manage your Word security settings to allow this macro to run.
	14. Select Start>Run , type REGEDI T, and then click OK ; the Registry Editor opens.
	You can also create a registry import file, as described in the sample code below.
	15. Create a name-value pair of type REG_SZ below the following registry key: HKEY_CURRENT_USER\Software\Mi crosoft\Offi ce \Word\Addi ns\ DM_COM_Addi n. WordAddi n.
	16. Name the name-value pair EventBeforeFooter. This name is case-specific. The value of the name-value pair is the name of the module you created, followed by a period, which is then followed by the name of the subroutine you created. The module name and the subroutine name are also case-specific.
	17. Ensure that all instances of Word are closed.
	18. Restart DM.
	19. Open Word, create a document, and save it to DM.
	20. From the Word menu, select Insert>Footer ; your custom footer information will appear.
Microsoft Word Customized Footer Sample Code	The following code example demonstrates how you can customize the footer with COM Automation. This code resides in a module named moduleDMEventCode , which is located in a Word template file named footer.dot , located in the user's Word Startup directory.
	' FILE NAME: footer.dot ' FILE LOCATION: per-use Word Startup directory ' MODULE NAME: moduleDMEventCode

- ' Purpose:
- $^{\prime}$ This module defines a custom subroutine that is called by

' DM when the user selects <code>Insert>DM</code> Footer from the Word menu.

 $^{\prime}$ The subroutine inserts customized information into the footer.

' To make the following code work, place the following 4 lines

' of text (minus the comment (') marks) into a .reg file and

' run it.

'Windows Registry Editor Version 5.00

'[HKEY_CURRENT_USER\Software\Microsoft\Office\Word\Addins \DM_COM_Addin. WordAddin]

'"EventBeforeFooter"="modul eDMEventCode.subEventBeforeFoo ter"

Public Sub subEventBeforeFooter()

This subroutine is called by DM when the Insert>DM Footer menu item is clicked in Word. ' The DM code that normally inserts the footer is not called, ' i.e. this code replaces that code. ' Collect the information intended for the footer. Dim strFooterText As String strFooterText = get_footer_information() ' Insert the information into the footer. insert footer strFooterText End Sub Private Sub insert_footer(strFooterText As String) Dim footer_selection As Selection Dim initial_active_window_view_type As Integer initial_active_window_view_type = ActiveWindow. View. Type Application. EnableCancelKey = wdCancelDisabled If (ActiveWindow.View.SplitSpecial = wdPaneNone) Then ActiveWindow. ActivePane. View. Type = wdPageView Else ActiveWindow. View. Type = wdPageView End If Acti veWindow. View. SeekView = wdSeekMainDocument ActiveWindow. View. SeekView = wdSeekPrimaryFooter ActiveDocument. Sections(1). Footers(wdHeaderFooterPrima ry)._ Shapes. Sel ectAl I Set footer selection = ActiveWindow. Selection ' Set the footer text to the sole parameter of this function. footer_selection.Text = strFooterText ActiveWindow. View. Type = wdPageView ActiveDocument. Sections(1). Footers(wdHeaderFooterPrima ry)._ Range = footer_selection. Range

footer_sel ection. Paragraphs. Alignment = wdAlignParagraphLeft ActiveWindow. View. Type = wdNormal View ActiveWindow. View. Type = initial_active_window_view_type Word.StatusBar = "" Application. EnableCancel Key = wdCancel Interrupt End Sub Private Function get_footer_information() As String Dim strFooterText As String Dim obj DOCSObjectsDM As Object Dim strActiveDocumentFullName As String Dim out_strValue As String Dim strDocumentName As String Dim strAuthorID As String Dim strDocumentNumber As String Dim strAbstract As String Dim strForm As String Dim strTypeID As String Dim strCreationDate As String Dim strLastEditDate As String Dim strClientName As String Dim strMatterName As String Dim strClientID As String Dim strMatterID As String Dim strVersionNumber As String strActiveDocumentFullName = ActiveDocument.FullName Set obj DOCSObj ectsDM = CreateObj ect("DOCSObj ects. DM") obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "DOCNAME", out_strValue strDocumentName = out strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "AUTHOR ID", out strValue strAuthorID = out strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "DOCNUM", out strValue strDocumentNumber = out strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "ABSTRACT", out_strValue strAbstract = out_strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "TYPE_ID", out_strValue strTypeID = out_strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "CREATION_DATE", out_strValue strCreationDate = out_strValue

obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "LASTEDITDATE", out_strValue strLastEditDate = out strValue obi DOCSObi ectsDM. GetDocI nfo strActi veDocumentFul I Name, "CLIENT_NAME", out_strValue strClientName = out strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "CLIENT_ID", out_strValue strClientID = out strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "FORM", out strValue strForm = out_strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "MATTER_NAME", out_strValue strMatterName = out_strValue obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name, "MATTER ID", out strValue strMatterID = out strValue strVersionNumber = _ get version number(strActiveDocumentFullName) strFooterText = "" ' Comment out any of the following lines to remove ' information from the footer. ' You may also re-order the lines below. ' BEGIN LINES WHICH MAY BE COMMENTED OUT OR RE-ORDERED strFooterText = strFooterText & "Document Name: " & _ strDocumentName & vbNewLine strFooterText = strFooterText & "Document Number: " & strDocumentNumber & vbNewLine strFooterText = strFooterText & "Version: " & strVersionNumber & vbNewLine strFooterText = strFooterText & "Author: " & strAuthorID & _ vbNewLi ne strFooterText = strFooterText & "Abstract: " & strAbstract & vbNewLi ne strFooterText = strFooterText & "Form: " & strForm & vbNewLine strFooterText = strFooterText & "TypeID: " & strTypeID & _ vbNewLi ne strFooterText = strFooterText & "Creation Date: " & _ strCreationDate & vbNewLine

strFooterText & "Last Edit Date: " & _

```
strLastEditDate & vbNewLine
   strFooterText = strFooterText & "Client Name: " & _
       strClientName & vbNewLine
   strFooterText = strFooterText & "Client ID: " &
   strClientID
       & vbNewLine
   strFooterText = strFooterText & "Matter Name: " & _
       strMatterName & vbNewLine
   strFooterText = strFooterText & "Matter ID: " &
   strMatterID _
      & vbNewLine
   ' END LINES WHICH MAY BE COMMENTED OUT OR RE-ORDERED
   get_footer_information = strFooterText
End Function
Private Function get_version_number _
   (strActiveDocumentFullName As String) As String
   Dim strVersionNumber As String
   Dim obj RegExp As Object
   Dim obj Matches, obj Match As Object
   Dim strMatchValue As String
   strVersionNumber = ""
   Set obj RegExp = CreateObj ect("VBScript.RegExp")
   obj RegExp. Gl obal = True
    The following regular expression looks for the
     following pattern:
     1 non-al phanumeric character
        followed by
     1 letter 'v', either upper or lower case
        followed by
     1 or more decimal digits
        followed by
     1 non-alphanumeric character
     Examples of matching patterns:
      _v1_
      _V11_
      -V23-
      -v17-
     Thus, it catches the version number regardless of the
     delimiter used to delimit
   ' the various parts of a DM file name.
   obj RegExp. Pattern = "[^a-zA-ZO-9][vV][0-9]+[^a-zA-ZO-
   91
   Set obj Matches =
   obj RegExp. Execute(strActiveDocumentFullName)
    The matches count will normally be one. For it to be
   ' different would generally require that the DM
   Admini strator
     had made a change to how DM names files.
   If obj Matches. Count = 0 Then
       strVersi onNumber = "<UNDEFINED>"
```

```
22 CHAPTER 1
```

```
' By default, and under normal circumstances, their
   ' should be only one match object.
                                       The exceptions
   will
   ' be in those situations where the author's id, the
   ' library name, or the document name (or some other
   ' value added to the document naming scheme by the
   ' DM Administrator) contains a string that matches
   ' the regular expression above (e.g. where the
   •
     document name is "automobile_v8_engines"). If
   thi s
   ' is an issue, perform further expression testing
   to ensure
   ' that this function returns the version number.
   Set obj Match = obj Matches. Item(0)
   strMatchValue = objMatch. Value
   strVersionNumber = Left(strMatchValue,
   Len(strMatchValue) - 1)
   strVersionNumber = Right(strVersionNumber,
   Len(strVersionNumber) _____
        - 2)
End If
get_version_number = strVersionNumber
```

```
End Function
```

El se

Using Word's AutoRecover Feature

If a system failure occurs and Word autorecovers the document, the file will be saved as a DM document.

- Word 2003 or XP: When you start Word after the system failure, you can choose to open the last-saved document or the autorecovered document.
- Word 2000: When you start Word after the system failure, the autorecovered document is opened.

If you choose to save the recovered document, you can select from one of the following options:

- Replace the existing last-saved document.
- Save the document as a new document.
- Save the document as a new version or subversion.

In order for the Word AutoRecover feature to work properly, you must disable the DM Auto-Recovery function in the the following registry key:

[HKEY_CURRENT_USER\Software\Hummi ngbi rd\PowerDOCS\Cor e\

Pl ugi ns\Fusi on\OM\Excl udes]

Ensure that the following value is set:

"MS WORD"=dword:0000001

Troubleshooting COM Automation for Word

	Should troubleshooting be necessary, a log file can be created in the DM Extensions directory after you enable the log file in the following registry key:	
HKEY_LOCAL_MACHINE\SOFTWARE\Hummingbird\Hummingbir DM Application Integration\Installation\Microsoft Create a new DWord value with the following values:		
	Name: Debug	
	Value: 2	
Word Stops Responding	Ensure that Microsoft Office SP3 is installed for Word 2000; otherwise, Word will stop responding when you open documents from DM.	
Auto-Recovery	Auto-Recovery integration is enabled for Word XP and later versions. If you want to ensure that this feature is working, open an already-saved document in DM and then open Windows Task Manager to end the winword.exe process.	
Previous Integration Files Must Be Removed	Ensure that previous DM integration files are removed from Word's Startup directory before using COM Automation; otherwise, both integration files will be loaded and the integration will fail.	
COM Automation for Word Registry Setting	The following registry entry is made when the DLL is registered: HKEY_CURRENT_USER\Software\Mi crosoft\Offi ce\Word\ Addi ns\Offi ce_COM_Addi n. WordAddi n	
	To remove integration, delete the Offi ce_COM_Addi n. WordAddi n registry key.	
E-mail Integration	When you use Word as the e-mail editor, you will be able to use all the DM menus and functions that are available in Active Word integration.	
	DM does not maintain a link with the e-mail document, so if you save the e-mail to DM, the document is uploaded and available immediately. If you edit the document and select File>Save , a new Profile form will appear.	
	Some e-mails cannot be saved to DM as they have HTML or Rich Text Format (RTF) format types. If you select File>Save As , a native dialog box	

will appear; however, if you click the **Reply** button, then Word starts and the e-mail can be saved to DM.

Saving to the Drafts Folder

If you create a new e-mail and then decide to save it before sending it by choosing **File>Save** to save it to Drafts folder, a Profile form appears instead. In order to save it to the Drafts folder you have to select **File>Close**.

2

Using COM Automation for Microsoft Excel

About COM Automation for Microsoft Excel

COM Automation for Microsoft Excel uses a Microsoft Office COM Add-in to provide active and passive integration with DM. This add-in is meant to remove the dependency on Microsoft's built-in ODMA implementation. The add-in communicates with DM through DMmanaged COM interfaces and has no reliance on ODMA.

If you are using any customized items for Microsoft Excel (templates, VBA macros, Excel add-ins), third-party add-ons, or other applications that have a dependency on the Excel ODMA implementation, then you should contact Technical Support before using the Excel Active COM Add-in integration.

This chapter contains the following sections:

- Installing COM Automation for Microsoft Excel
- Customizing the DM Footer with COM Automation
- Using Excel's Auto-Recovery Feature
- Troubleshooting COM Automation for Excel

Supporting Software

COM Automation for Microsoft Excel is supported with DM 5.1.0.5 Service Release 5 and later.

This feature is supported for the following versions of Microsoft Excel:

- Excel 2000 SP3
- Excel XP SP3
- Excel 2003 SP2

New Functionality

- The **XClose** command is available, which means you can close Excel by clicking the **X** (Close) button.
- The **Ctrl+S** keystroke combination saves correctly when in edit cell mode.
- A new **Insert>Footer** menu option is added for Active integration. A new **DM>Footer** menu option has been added for Passive integration.

Refer to Customizing the DM Footer with COM Automation for more information about creating Excel footers.

• Excel's Auto-Recovery feature has been integrated so that a Save As Options dialog box appears if Excel stops responding when you save a workbook. Consult the Using Excel's Auto-Recovery Feature section for more information.

Installing COM Automation for Microsoft Excel

Create a Launch Method

You can create a launch method through DM Webtop or by using the Library Maintenance tool, which is accessible on the DM server computer or any computer where the Server Admin Tools are installed.

Creating a Launch Method in Library Maintenance

Following are instructions on how to create the launch method in Library Maintenance.

To create a launch method in Library Maintenance:

- 1. Click Start>Programs>Hummingbird>DM Server>Server Admin Tools>Library Maintenance.
- 2. Click Applications.
- 3. Double-click MS EXCEL.
- 4. In the Application Maintenance window, click Launch Methods.
- 5. In the Application Launch Methods window, click Copy.
- 6. Enter the following information:

Enabled: Ensure that this check box is selected.

Description: Enter a description for the launch method, such as COM Automation for Microsoft Excel (Active/Passive).

Location: Ensure that the location to the **excel.exe** executable file is specified.

Command Line Parameters: Leave the **Command Line Parameters** field blank.

Default Directory: Leave this field blank.

Search Mapping: Leave this field blank.

Integration: Select Full Integration.

- 7. Click **Apply**; the **DDE Settings** button becomes available.
- 8. Click **DDE Settings**; ensure that the DDE settings match the settings specified below:

Open Settings:

Application Name EXCEL

Topic Name	SYSTEM
Command	[RUN("'DDEOpen ""%1""")]
Command (not running)	[RUN("'DDEOpen ""%1""")]

Print Settings:

Application Name	EXCEL
Topic Name	SYSTEM
Command	[RUN("'DDEPrint ""%1"", ""1""")]
Command (not running)	[RUN("'DDEPrint ""%1"", ""0""")]

- 9. Click Close.
- 10. Click **OK** in the Application Launch Methods window.
- 11. Click Close.
- 12. Click the **Options** button from the Application Maintenance dialog box.
- 13. Ensure that the **Shows on Desktop** and **Valid on Profile** check boxes are selected; click **OK**.
- 14. Click **OK**; the launch method for COM Automation for Excel is now complete.

Creating a Launch Method in DM Webtop

Following are instructions on how to create a launch method in DM Webtop.

To create a launch method in DM Webtop:

- 1. Click DM Admin>Validation Table>Applications.
- 2. Click **MS EXCEL** to display the settings for Microsoft Excel.
- 3. Click Launch Methods.
- 4. Click **Add**. (Alternatively, you can click **Copy** and modify an existing launch method.)
- 5. In the **Description** field, enter Microsoft Excel COM Automation (Active/Passive Integration).
- 6. In the **Location** field, enter the path where the **excel.exe** executable is located.

7. Enter the following launch method settings:

Command Line Parameters	Leave this field blank.
Integration Type	Full Integration

- 8. Select the **Enabled** check box.
- 9. Select **DDE Settings**. You will be prompted to save the changes to the launch method.
- 10. Enter the DDE Settings as follows:

Open Settings:

Application Name	EXCEL
Topic Name	SYSTEM
Command	[RUN("'DDEOpen ""%1""")]
Command (not running)	[RUN("'DDEOpen ""%1""")]

Print Settings:

Application Name	EXCEL
Topic Name	SYSTEM
Command	[RUN("'DDEPrint ""%1"", ""1""")]
Command (not running)	[RUN("'DDEPrint ""%1"", ""0""")]

- 11. Click **Save** and **Close** to complete creation of the launch method for COM Automation for Microsoft Excel.
- 12. Select **Save** again to save the application settings for Microsoft Excel.

Refresh DM Server Caches

fver After you create the launch method for COM Automation for**thes** Microsoft Excel, you must refresh the DM Server caches. On the DM Server machine, go to the Server Manager's **Caches** tab and select **Refresh** All.

Implement COM Automation for Excel on User Workstations

You can install the COM Automation Add-in for Excel from DM Webtop or by using the DM Extensions Server Setup. Following are instructions for both methods.

Using DM Extensions Server Setup

To implement COM Automation for Microsoft Excel in DM Extensions Server Setup:

- 1. Click **Start>Programs>Hummingbird>DM Extensions Server Setup>Install DM Extensions**; the Installation Wizard appears and the installation begins.
- 2. Accept the default location specified to install the media, or click **Browse** to enter an alternation location; click **Next**.

If you have already installed DM Extensions, select the **Modify** option, click **Next**, and skip to step 4 below.

- 3. In the Choose Destination Location window, accept the default Destination Folder in which to install DM Extensions, or click the **Browse** button to specify an alternate location; click **Next**.
- 4. The Select Optional Components window appears; click the **DM Application Integration** plus (+) sign to expand the tree.
- 5. Expand the Microsoft Applications tree.
- 6. Expand the Microsoft Excel tree.
- 7. Expand the Microsoft Excel Active Integration tree.
- 8. Expand the via COM Automation tree.
- 9. Select one of the following check boxes:
 - with Front-End Profiling
 - with Front-End Profiling and Cost Recovery
 - Active Integration without Front-End Profiling

If you want to install COM Automation using Passive integration, in step 7, expand the **Microsoft Excel Passive Integration** tree, and then click the **via COM Automation** check box.

- 10. Click Next.
- 11. In the Connection Information window, enter then machine name of the DM Server, or click **Browse** to locate the DM Server; then click **Next**.

- 12. The Start Copying Files window appears; click **Next** to start the installation.
- 13. When the installation is complete, click **Finish**.

Using DM Webtop

To implement COM Automation for Microsoft Excel in DM Webtop:

- 1. In DM Webtop, click My Options.
- 2. Click the **Optional Components** tab.
- 3. Click the **Install** icon; the Installation Wizard appears and the installation begins.
- 4. Accept the default location specified to install the media, or click **Browse** to enter an alternation location; click **Next**.

If you have already installed DM Extensions, select the **Modify** option, click **Next**, and skip to step 6 below.

- In the Choose Destination Location window, accept the default Destination Folder in which to install DM Extensions, or click the **Browse** button to specify an alternate location; click **Next**.
- 6. The Select Optional Components window appears; click the **DM Application Integration** plus (+) sign to expand the tree.
- 7. Expand the Microsoft Applications tree.
- 8. Expand the **Microsoft Excel** tree.
- 9. Expand the Microsoft Excel Active Integration tree.
- 10. Expand the via COM Automation tree.
- 11. Select one of the following check boxes:
 - with Front-End Profiling
 - with Front-End Profiling and Cost Recovery
 - Active Integration without Front-End Profiling

If you want to install COM Automation using Passive integration, in step 9, expand the **Microsoft Excel Passive Integration** tree, and then click the **via COM Automation** check box.

- 12. Click Next.
- 13. If the Connection Information window appears, enter then machine name of the DM Server, or click **Browse** to locate the DM Server, and then click **Next**.

- 14. The Start Copying Files window appears; click **Next** to start the installation.
- 15. When the installation is complete, click **Finish**.

Customizing the DM Footer with COM Automation

You might have customized Excel footers based on DM's macro functions, which are built upon ODMA. Because COM Automation does not use ODMA, these particular customizations will no longer work; however, you can customize Excel footers with COM Automation. You should be familiar with Visual Basic for Applications (VBA) and the Microsoft VBA editor before performing the following steps.

To create a custom Excel footer using COM Automation:

- 1. Open Microsoft Excel.
- 2. Open the Microsoft VBA editor by pressing ALT+F11 or by selecting Tools>Macros>Visual Basic Editor.
- 3. Select **File>Save** and save the document locally as an Excel Add-in (*.xla). The name should not conflict with any other macro file names.
- 4. In the VBA Project Explorer, select the project containing the current workbook.
- 5. Right-click the **Modules** node below this project and select **Insert>Module**.
- 6. In the **Properties** window, rename the module. Ensure that the file name does not conflict with any other file names.
- 7. Add a **Public** subroutine; the name of the subroutine can be anything that VBA allows and does not conflict with other names. The subroutine must take only one argument and that argument must be of the type workbook. The name of the argument can be anything allowed by VBA that does not conflict with other variable names.
- 8. In this subroutine, add your custom code that will insert the desired information into the footer. The process of inserting the footer information involves calls to the Microsoft Excel object model. Depending on what you want inserted into the footer, the subroutine might also

contain calls to the DM API, the DM Extensions API, other
third-party APIs, and VBA functions. Note that the footer
must be less than 254 characters in length; this is a
restriction imposed by Excel. Refer to Microsoft Excel
Customized Footer Sample Code for more information.

- 9. From the VBA editor menu, select File>Save.
- 10. Select File>Close; focus returns to Excel.
- 11. From the Excel menu, select File>Exit.
- Copy the .xla file to one of the Excel Startup directories, such as C:\Program Files\Microsoft
 Office\OFFICE11\XLSTART. You must manage your Excel security settings to allow this macro to run.
- 13. Select **Start>Run**, type REGEDI T, and then click **OK**; the Registry Editor opens.
- 14. Create a name-value pair of type REG_SZ below the following registry key:
 HKEY_CURRENT_USER\Software\Mi crosoft\Offi ce \Excel \Addi ns\DM_COM_Addi n. Excel Addi n.
- 15. Name the name-value pair EventBeforeFooter. This name is case-specific. The value of the name-value pair is the name of the module you created, followed by a period, which is then followed by the name of the subroutine you created. The module name and the subroutine name are also case-specific.
- 16. Open Excel, create a document, and save it to DM.
- 17. From the Excel menu, select **Insert>Footer**; your custom footer information will appear.

el The following code demonstrates how you can customize the footer with COM Automation. This code resides in a module named moduleDMEventCode, which is located in an Excel template file named footer.xla, located in the user's Excel Startup directory.

' FILE NAME: footer.xla
' FILE LOCATION: an Excel Startup directory
' MODULE NAME: moduleDMEventCode
' Purpose:
' This module defines a custom subroutine that is called by

Microsoft Excel Customized Footer Sample Code ' DM when the user selects <code>Insert>DM</code> Footer from the <code>Excel</code> menu.

 $^{\prime}$ The subroutine inserts customized information into the footer.

' To make the following code work, place the following 4 lines

' of text (minus the comment (') marks) into a .reg file and

' run it.

'Windows Registry Editor Version 5.00

' [HKEY_CURRENT_USER\Software\Microsoft\Office\Excel \Addin s\DM_COM_Addin. Excel Addin]

'"EventBeforeFooter"="modul eDMEventCode. subEventBeforeFoo ter"

Public Sub subEventBeforeFooter(objWorkbook As Workbook)
Dim strFooterText As String
' You may insert any footer information.

strFooterText = get_footer_information()

Dim objWorksheet As Worksheet

Set objWorksheet = objWorkbook.ActiveSheet

' Excel limits the length of the footer.

strFooterText = Left(strFooterText, 250)

objWorksheet.PageSetup.LeftFooter = strFooterText
End Sub

Private Function get_footer_information() As String Dim strFooterText As String Dim objDOCSObjectsDM As Object Dim strActiveDocumentFullName As String Dim out strValue As String Dim strDocumentName As String Dim strAuthorID As String Dim strDocumentNumber As String Dim strAbstract As String Dim strForm As String Dim strTypeID As String Dim strCreationDate As String Dim strLastEditDate As String Dim strClientName As String Dim strMatterName As String Dim strClientID As String Dim strMatterID As String Dim strVersionNumber As String strActiveDocumentFullName = ActiveWorkbook.FullName Set obj DOCSObj ectsDM = CreateObj ect("DOCSObj ects. DM") obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,

"DOCNAME", out_strValue

```
strDocumentName = out_strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "AUTHOR_ID", out_strValue
strAuthorID = out strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "DOCNUM", out_strValue
strDocumentNumber = out strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "ABSTRACT", out strValue
strAbstract = out strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "TYPE_ID", out_strValue
strTypeID = out strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "CREATION_DATE", out_strValue
strCreationDate = out strValue
obi DOCSObi ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "LASTEDITDATE", out strValue
strLastEditDate = out_strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "CLIENT_NAME", out_strValue
strClientName = out_strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "CLIENT ID", out strValue
strClientID = out strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
"FORM", _
   out strValue
strForm = out_strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "MATTER_NAME", out_strValue
strMatterName = out_strValue
obj DOCSObj ectsDM. GetDocI nfo strActi veDocumentFul I Name,
   "MATTER_ID", out_strValue
strMatterID = out_strValue
strVersionNumber =
   get_versi on_number(strActi veDocumentFul | Name)
strFooterText = ""
' Comment out any of the following lines to remove
' information from the footer.
' You may also re-order the lines below.
' BEGIN LINES WHICH MAY BE COMMENTED OUT OR RE-ORDERED
strFooterText & "Document Name: " & _
```

```
strDocumentName & "; "
   strFooterText = strFooterText & "Document Number: " &
       strDocumentNumber & "; "
   strFooterText = strFooterText & "Version: " &
   strVersionNumber _
      & ": "
   strFooterText = strFooterText & "Author: " &
   strAuthorID _
      & " : "
   strFooterText = strFooterText & "Abstract: " &
   strAbstract _
      & " : "
   strFooterText = strFooterText & "Form: " & strForm & ";
   strFooterText = strFooterText & "TypeID: " & strTypeID
   & ";
   strFooterText = strFooterText & "Creation Date: " & _
      strCreationDate & "; "
   strFooterText = strFooterText & "Last Edit Date: " & _
       strLastEditDate & "; "
   strFooterText = strFooterText & "Client Name: " & _
      strClientName & "; "
   strFooterText = strFooterText & "Client ID: " &
strClientID & "; "
   strFooterText = strFooterText & "Matter Name: " & _
      strMatterName & "; "
   strFooterText = strFooterText & "Matter ID: " &
   strMatterID
   ' END LINES WHICH MAY BE COMMENTED OUT OR RE-ORDERED
   get_footer_information = strFooterText
End Function
Private Function get_version_number _
   (strActiveDocumentFullName As String) As String
   Dim strVersionNumber As String
   Dim obj RegExp As Object
   Dim obj Matches, obj Match As Object
   Dim strMatchValue As String
   strVersionNumber = ""
   Set obj RegExp = CreateObj ect("VBScript.RegExp")
   obj RegExp. Gl obal = True
   ' The following regular expression looks
   ' for the following pattern:
     1 non-alphanumeric character
        followed by
     1 letter 'v', either upper or lower case
        followed by
   ' 1 or more decimal digits
        followed by
     1 non-al phanumeric character
    Examples of matching patterns:
```

v1

- V11
- ' -V23-

' -v17-

' Thus, it catches the version number regardless of

```
' the delimiter used to delimit the various parts of
```

' a DM file name.

objRegExp.Pattern = "[^a-zA-ZO-9][vV][0-9]+[^a-zA-ZO-9]"

Set obj Matches =

obj RegExp. Execute(strActiveDocumentFullName)

' The matches count will normally be one. For it to be

' different would generally require that the DM

' Administrator had made a change to how DM names files.

If obj Matches. Count = 0 Then

strVersionNumber = "<UNDEFINED>"

El se

' By default, and under normal circumstances, their should

' be only one match object. The exceptions will be in those

' situations where the author's id, the library name, or the

' document name (or some other value added to the document

' naming scheme by the DM Administrator) contains a string

' that matches the regular expression above (e.g. where the

' document name is "automobile_v8_engines"). If this is an

' issue, perform further expression testing to ensure that

' this function returns the version number.

```
Set obj Match = obj Matches. Item(0)
```

```
strMatchValue = objMatch.Value
```

```
strVersionNumber = Left(strMatchValue, _
```

```
Len(strMatchValue) - 1)
```

strVersionNumber = Right(strVersionNumber, _

Len(strVersionNumber) - 2)

End If

```
get_versi on_number = strVersi onNumber
```

```
End Function
```

Using Excel's Auto-Recovery Feature

For Auto-Recovery to work properly, you must disable DM Auto-Recovery through the following registry key:

[HKEY_CURRENT_USER\Software\Hummi ngbi rd\PowerDOCS\Cor e\

Pl ugi ns\Fusi on\OM\Excl udes]

Ensure that the following value is set:

"MS EXCEL"=dword:0000001

Excel 2000 does not support Auto-Recovery.

Troubleshooting COM Automation for Excel

Should troubleshooting be necessary, a log file can be created in the DM **Extensions** directory after you enable the log file in the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Hummingbird\Hummingbird DM Application Integration\Installation\Microsoft Add a **DWord** value with the following values:

Name: Debug

Value: 2

Previous Integration Files Must Be Removed Ensure that previous DM integration files are removed from Excel's **Startup** directory before using COM Automation; otherwise, both integration files will be loaded and the integration will fail.

3

Using Microsoft Office Menu Shortcut Keys with COM Automation

When you integrate Microsoft Word and/or Excel with DM using COM Automation with Active integration, some menu shortcut keys are intercepted and DM dialog boxes and controls are used instead of native functionality. For example, if you create a new document and select **File>Save** (**CTRL+S**), a Document Profile form appears for you to complete instead of the native Save dialog box.

However, when you integrate Word and/or Excel using COM Automation with Passive integration, the shortcut keys behave differently. For example, when you save a new or previously unmanaged document to DM, the **File>Save** (**CTRL+S**) shortcut key opens the native Save dialog box instead of displaying a Profile form. To save the document to DM, you must select **DM>Save As** or assign a different shortcut key combination to the DM command.

The tables shown on following pages list the default key mappings that Word and Excel applications use to call its functions. To see all native key mappings, select **Tools>Customize>Commands**, and then click the **Keyboard** button. See the following image to see an example of the key mappings for **File>Save** in Word.

Customize Keyboard	×
Specify a command	he
Categories:	Commands:
File Cit	FilePrint FilePrintPreview FilePrintSetup
Insert Format	FileProperties
Tools Table	FileSaveAll FileSaveAs
Specify keyboard sequence	
Current keys:	Press new shortcut key:
Ctrl+S A Shift+F12 Alt+Shift+F2	
Sa <u>v</u> e changes in:	Normal.dot
Description	
Saves the active document	or template
Assign Remove	Reset All Close

Once DM reads these key mappings and maps them to DM functions, you will not see these mappings in the Customize Keyboard dialog box as shown above. To see the default list, you must view this dialog box in a non-integrated version of Word.

Microsoft Office Intercepted Shortcut Keys

Word Intercepted Shortcut Keys

Word Menu Action	Intercepted Shortcut Keys
File>Save	CTRL+S
	SHIFT+F12
	ALT+SHIFT+F2
File>SaveAs	F12

Word Menu Action	Intercepted Shortcut Keys
File>Open	CTRL+O
	CTRL+F12
	ALT+CTRL+F2
File>Print	CTRL+P
	CTRL+SHIFT+F12

Excel Intercepted Shortcut Keys

Excel Menu Action	Intercepted Shortcut Keys
File>Save	CTRL+S SHIFT+F12
File>SaveAs	F12
File>Open	CTRL+O CTRL+F12
File>Close	CTRL+W CTRL+F4

Removing DM Menu Items

In some instances, you might want to remove a DM menu item such as **File>Save With Options**. Knowledge of Microsoft VBA is required to do this. Please refer to the following code example.

```
Sub RemoveDMmenu(menultem As String)
Dim cmdBar As CommandBar
Dim ctrl As CommandBarControl
Set cmdBar = Application.CommandBars("Menu Bar")
Set ctrl = cmdBar.FindControl(Tag: =menultem,
recursive: =True)
If (Not (ctrl Is Nothing)) Then
ctrl.Delete
End If
```

End Sub

The sub would be called, passing in a tag string. The example below removes the Save With Options function:

Call RemoveDMmenu("SWO")

DM Intercepted Menu Items

The DM-intercepted menu items are listed below with their assigned tag name.

Active Integration

Active Integration Menu Item	Tag Name
File>Save With Options	SWO
Tools>Compare And Merge	MERGE
File>Close Workspace	SWSCLOSE
Insert>Footer	FOOTER
Insert>File	INSERTFILE
Help>About Hummingbird DM	ABOUT

Passive Integration

Passive Integration Menu Item	Tag Name
DM>Open	DMOpen
DM>Save As	DMSaveAs
DM>Save With Options	DMSWO
DM>Close Workspace	DMSWOCLOSE
DM>Insert File	DMInsertFile
DM>Insert Picture	DMInsertPic
DM>Insert Footer	DMFooter
DM>Compare and Merge	DMMerge
DM>About Hummingbird DM	DMAbout
DM>Save Workspace	DMSWORKSPACE

4

DM Simplified API Technical Information

DM Simplified API

The DM Simplified API hides the complex standard application integration operations made via the DM Extensions API, and provides an interface in which inhouse and third-party developers can rapidly develop DM integration for third-party applications. DM Simplified API contains the more commonly-used functionality of the DM Extensions API.

Technical Specifications

DM Simplified API refers to the DM object located in the DOCSFusion Document Management Objects and Services type library, such as the **Fsplugin.dll** file. The DM object exposes an automation interface with seven methods and one property.

Methods

The methods are: **OpenDoc**, **CloseDoc**, **SaveDoc**, **SaveDocAs**, **NewDoc**, **IsDMDoc**, and **GetDocInfo**.

Property

The property is Library.

Functions

DM.OpenDoc The **OpenDoc** function displays a Quick Retrieve window Based on the user's response, it will launch the associated application, and then open the document in it via DDE.

Function OpenDoc(bstrFilePath As String, [bSingleSelect As Boolean], [bstrApplD As String]) As IntegerParameters bstrFilePath

[out] On success, this parameter will be assigned the path of the document that is opened. On failure, this parameter will be an empty string.

bSingleSelect

[optional] Used to specify whether or not multiple selections can be made on the Quick Retrieve screen. The default value is **FALSE (0)**.

bstrAppID

[in, optional] Used to specify a (compatible) Application ID for the document to be opened into. The default value is **NULL**.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1) and the document will be opened in the associated application. If the user cancels the Quick Retrieve dialog box, the return value will be VB_FALSE (0). Conversely, if the user cancels the Quick Retrieve dialog box and the library setting **Disable Native Open/Save** is not selected, the return value will be APP_SELECT (8). This should indicate to the calling application that a native Open File dialog box should be displayed. If the user is presented with the DM logon dialog box during this call and selects **Cancel**, the return code will be NOT_LOGGED_I N (10).

Remarks

If **bSingleSelect** is TRUE (1), only one document in the Quick Retrieve list can be selected, and the function will return the path to the download document without opening the document into the associated application. This parameter can be useful for creating mail merges and attachments.

DM.CloseDoc The CloseDoc function is used to release DM's reference to the opened document object.

Function CloseDoc(bstrFilePath As String) As Integer

Parameters

bstrFilePath

[in] Path of the document to be released.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1), and the document's status will be changed to Available. If the function fails, the return value will be VB_FALSE (0). If the user is presented with the DM logon dialog box during this call and selects **Cancel**, the return code will be NOT_LOGGED_IN (10).

Remarks

The **CloseDoc** function will not cause an application to close the actual file.

DM.SaveDoc The **saveDoc** function will upload the file specified in the **bstrFilePath** variable to the document server.

Function SaveDoc(bstrFilePath As String) As Integer

Parameters

bstrFilePath

[in] Path of the document to be saved to DM.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0). If the document is not currently opened by DM (or is not a DM-managed document), the return value will be UNMANAGED (9). A return value of UNMANAGED should indicate to the calling application that **DM.SaveDocAs** should be called in order to save the document to DM. If the user is presented with the DM logon dialog box during this call and selects **Cancel**, the return code will be NOT_LOGGED_IN (10).

DM.SaveDocAs The **SaveDocAs** function will save a new or existing document into DM.

Function SaveDocAs(bstrFilePath As String, bstrNewFilePath As String, [bstrAppID As String], [in, optional] bAutoRecovered As Boolean) As Integer

Parameters

bstrFilePath

[in] Path of the document to be saved to DM.

bstrNewFilePath

[in, out] Will contain the path of the new version or document if created from an existing version. If the user selected **Cancel** or **Replace Original**, an empty string will be returned.

bstrAppID

[in, optional] Can be used to prefill the **Application** field of the profile form. The default value is **NULL**.

bAutoRecovered

[in, optional] Specifies whether or not the current document has been auto-recovered by an application. Passing VB_TRUE (-1) will allow DM to reassociate the Checked-Out status of the profile with current SAPI operations. The default value is **VB_FALSE (0)**.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0). If the user cancels the Profile form dialog box and the library setting **Disable Native Open/Save** is not selected, the return value will be APP_SELECT (8). This should indicate to the calling application that a native **File Save** dialog box should be displayed. If the user is presented with the DM

logon dialog box during this call and selects $\mbox{Cancel},$ the return code will be NOT_LOGGED_I N (10).

DM.NewDoc The NewDoc function will create a new DM document based on an existing DM template.

Function NewDoc(bstrFilePath As String) As Integer

Parameters

bstrFilePath

[in, out] Path of the document created by DM.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0). If the user is presented with the DM logon dialog box during this call and selects **Cancel**, the return code will be NOT_LOGGED_IN (10).

DM.ISDMDOC The **ISDMDoc** function will determine if a document is currently opened in DM.

Function IsDMDoc(bstrFilePath As String) As Integer

Parameters

bstrFilePath

[in] Path of the document to be evaluated.

Return Values

If the document is a DM document, the return value will be VB_TRUE (-1). If the document is not a DM document or the DM document's status is not **Checked Out**, the return value will be VB_FALSE (0). If the user is presented with the DM logon dialog box during this call and selects **Cancel**, the return code will be NOT_LOGGED_IN (10).

DM.GetDocInfo The GetDocInfo function will retrieve profile information relating to the opened DM document.

Function GetDocInfo(*bstrFilePath As String, bstrColumnName As String, bstrColumnValue As String*) As Integer

Parameters

bstrFilePath

[in] Path of the document to be evaluated.

bstrColumnName

[in] The name of the target profile column.

bstrColumnValue

[out] The value of the profile column corresponding to **bstrColumnName**.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1) and **bstrColumnValue** will contain a string. If the function fails, the return value will be VB_FALSE (0) and **bstrColumnValue** will contain an empty string. If the user is presented with the DM logon dialog box during this call and selects **Cancel**, the return code will be NOT_LOGGED_IN (10).

DM.Library The read only **Library** property will contain the name of the library which the user is logged on to.

Property Library As String

Remarks

If the user is currently logged on to DM, this property will contain the name of the logged-on library. If the user is not logged on, this property will contain an empty string. This property will not cause the DM logon dialog box to be displayed if the user is not currently logged on. This is an excellent way to determine whether or not a user is currently logged on to DM.

Sample Code

The sample code shown below was written in Microsoft Word VBA. To use these sample routines, you must include DOCSFusion Document Management Objects and Services type library as a reference in your VB or VBA project.

DM.OpenDoc	'Declarations Dim appInt As DOCSObjects.DM Const APP_SELECT = 8 Const UNMANAGED = 9 Const NOT_LOGGED_IN = 10
	'Return Codes = True, False, APP_SELECT, NOT_LOGGED_IN Sub SimpleApiOpen ()
	Dim filepath As String Dim ret As Integer Dim SingleSelect As Boolean
	SingleSelect = False
	If appInt Is Nothing Then Set appInt = CreateObject("DOCSObjects.DM") End If
	ret = applnt.OpenDoc(filepath, SingleSelect)
	Select Case ret Case True: If SingleSelect = True Then Documents.Open (filepath) End If Case False: 'user cancelled Case APP_SELECT: 'show native open dialog Application.Dialogs(wdDialogFileOpen).Show Case NOT_LOGGED_IN: 'User did not log in to DM End Select
	End Sub
DM.CloseDoc	'Return Codes = True, False, NOT_LOGGED_IN Sub SimpleApiClose ()

Dim filepath As String Dim ret As Integer If appInt Is Nothing Then Set appInt = CreateObject("DOCSObjects.DM") End If filepath = ActiveDocument.FullName ret = appInt.CloseDoc(filepath) Select Case ret Case True: ActiveDocument.Close False Case False: Case NOT_LOGGED_IN: 'User did not log in to DM End Select

End Sub

```
DM.SaveDoc
                           'Return Codes = True, False, UNMANAGED, NOT_LOGGED_IN
                           Sub SimpleApiSave ()
                               Dim filepath As String
                               Dim ret As Integer
                               If appInt Is Nothing Then
                                   Set appInt = CreateObject("DOCSObjects.DM")
                               End If
                               If Word. ActiveDocument. Path = "" Then
                                   ret = appInt.SaveDoc(filepath)
                               El se
                                   ActiveDocument. Save
                                   filepath = ActiveDocument.FullName
                                   ret = appInt.SaveDoc(filepath)
                               End If
                               Select Case ret
                                   Case True:
                                       ActiveDocument. SaveAs filepath
                                   Case False:
                                       MsgBox "User Cancel ed"
                                   Case UNMANAGED:
                                       MsgBox "Not a DM document"
                                   Case NOT_LOGGED_IN:
                           'User did not log in to DM
                               End Select
                           End Sub
```

```
DM.SaveAs
                         'Return Codes = True, False, APP_SELECT, NOT_LOGGED_IN
                         Sub SimpleApiSaveAs ()
                             Dim filepath As String
                             Dim NewFilePath As String
                             Dim ret As Integer
                             If appInt Is Nothing Then
                                 Set appInt = CreateObject("DOCSObjects.DM")
                             End If
                             filepath = ActiveDocument.FullName
                             ret = appInt.SaveDocAs(filepath, NewFilePath, "MS
                         WORD")
                             Select Case ret
                                 Case True:
                                     Application. ActiveDocument. SaveAs NewFilePath
                                 Case False:
                                 Case APP_SELECT:
                                     MsgBox "AppSelect"
                                 Case NOT_LOGGED_IN:
                         'User did not log in to DM
                             End Select
```

End Sub

	'Return Codes = True, False, NOT_LOGGED_IN Sub SimpleAPIIsDMDoc ()
	Dim ret As Integer
	<pre>If appInt Is Nothing Then Set appInt = CreateObject("DOCSObjects.DM") End If</pre>
ret = appInt.IsDMDoc(Active	<pre>ret = appInt.IsDMDoc(ActiveDocument.Path)</pre>
	Select Case ret
	Case True:
	MsgBox "ActiveDocument is a DM document"
	Case False:
	MsgBox "ActiveDocument is not a DM document"
Case NOT_LOGGED_IN: 'User did not log in to DM	Case NOT_LOGGED_IN:
	'User did not log in to DM
	End Select
	End Sub

```
DM.GetDocInfo
                              'Return Codes = True, False, NOT_LOGGED_IN
                              Sub SimpleAPIGetProfileInfo ()
                                  Dim filepath As String
                                  Dim ColValue As String
                                  Dim colat As String
                                  Dim ret As Integer
                                  If appInt Is Nothing Then
                                      Set appInt = CreateObject("DOCSObjects.DM")
                                  End If
                                  filepath = ActiveDocument.FullName
                                  ret = appInt.GetDocInfo(filepath, "DOCNAME", ColValue)
                                  colat = "Docname = " & ColValue & Chr(13)
                                  ret = appInt.GetDocInfo(filepath, "AUTHOR_ID",
                              Col Val ue)
                                  colat = colat & "Author = " & Col Value & Chr(13)
                                  ret = appInt.GetDocInfo(filepath, "ABSTRACT",
                              Col Val ue)
                                  colat = colat & "Typist = " & ColValue & Chr(13)
                                  Select Case ret
                                      Case True:
                                          MsgBox colat, vbOKOnly, "Hummingbird"
                                      Case False:
                                      Case APP_SELECT:
                                      Case NOT_LOGGED_IN:
                              'User did not log in to DM
                                  End Select
                              End Sub
```

5

DM COM Add-in Programmer's Guide for Microsoft Word

The DM COM Add-in exposes an automation interface with the following methods:

- DMClose
- DMCompare
- DMCompareAndMerge
- DMExit
- DMFooter
- DMInsertFile
- DMInsertPicture
- DMNativeSave
- DMNativeSaveAs
- DMOpen
- DMPasswordProtectDocument
- DMSave
- **DMPrint**—DMPrint is obsolete and is not used by the DM COM Add-in.

DM_COM_Addin.clsWord.DMClose

The **DMClose** function is used to release the DM reference to the opened document

object.

Function DMClose(oDoc As Document) As Boolean

Parameters

oDoc

[in] Word Document object.

Return Values

If the function succeeds, the return value will be VB_FALSE (0), and the document's status will be changed to Available. If the function fails, the return value will be VB_TRUE (-1).

Remarks

The **CloseDoc** function will not cause an application to close the actual file. **DMClose** is called by a Word event: **DocumentBeforeClose**.

DM_COM_Addin.clsWord.DMCompare

The **DMCompare** function is used to compare the active document to a DM document. Note that this function is used only in Word 2000.

Function dmCompare() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

DM_COM_Addin.clsWord.DMCompareAndMerge

The **DMCompareAndMerge** function is used to release the DM reference to the opened document object.

Function DMCompareAndMerge() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

This function is used in Microsoft Word XP and later versions.

DM_COM_Addin.clsWord.DMExit

The **DMExit** function is used to release the DM reference to all open documents, to close the actual files, and to exit Word.

Function DMExit() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

DM_COM_Addin.clsWord.DMFooter

The **DMFooter** function is used to add a footer to the document that contains DM profile data.

Sub DMFooter(oDoc As Document)

Parameters

oDoc

[in] Word Document object.

Remarks

This function inserts the Document Number, Document Name, Document Author, Document Type, Document Abstract, and Application ID.

DM_COM_Addin.clsWord.DMInsertFile

The **DMInsertFile** function is used to insert a DM document into the active document.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

The Microsoft Word document format converters must be installed before using this function.

DM_COM_Addin.clsWord.DMInsertPicture

The **DMInsertPicture** function is used to insert a DM picture into the active document.

Function DMInsertPicture() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

You must select a valid picture file, such as .JPG.

DM_COM_Addin.clsWord.DMNativeSave

The **DMNativeSave** function is used to save an existing DM or native document when using Passive Integration.

Function DMNativeSave(oDoc As Document) As Boolean

Parameters

oDoc

[in] Word Document object.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

DM_COM_Addin.clsWord.DMNativeSaveAs

The **DMNativeSaveAs** function is used to save a document as a new document to the local file system.

Function DMNativeSave(oDoc As Document) As Boolean

Parameters

oDoc

[in] Word Document object.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

If the document is a DM document, the document's status will be changed to Available.

DM_COM_Addin.clsWord.DMOpen

The **DMOpen** function is used to open documents from DM.

Function DMOpen() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

DM_COM_Addin.clsWord.DMPasswordProtect Document

The DMPasswordProtectDocument function is used to show the Save With Options dialog box.

Sub DMPasswordProtectDocument()

DM_COM_Addin.clsWord.DMPrint

The DM COM Add-in does not use this function.

DM_COM_Addin.clsWord.DMSave

The **DMSave** function is used to save and upload the document to DM.

Function DMSave(oDoc As Document) As Boolean

Parameters

oDoc

[in] Word Document object.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

If the document is not managed, the application will perform a native save.

DM_COM_Addin.clsWord. DMSaveAS

The **DMSaveAS** function is used to create the Document Profile and get a return path to save the actual file.

```
Function DMSaveAS(oDoc As Document, [sProtectPass As
String], [sWritePass As String], [ReadOnlyRec As Boolean],
[CalledFromSave As Integer], [CalledFromNew As Integer],
[fileType], [blsRecovered As Boolean = False]) As Boolean
```

Parameters

oDoc

[in] Word Document object.

sProtectPass

[in,optional] Protection password to apply to document.

sWritePass

[in,optional] Write-Reserve password to apply to document.

ReadOnlyRec

[in,optional] Open document as read-only recommended.

CalledFromSave [in,optional] Not used. CalledFromNew [in,optional] Not used. fileType [in,optional] File format of the document. blsRecovered

[in,optional] Document is Auto-Recovered.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

The **DMSaveAS** function does not upload the file.

Sample Code

The sample code below was written in Microsoft Word VBA. To use these sample routines, you must include the Office DM Integration Addin type library as a reference in your VB or VBA project.

DM COM Addia de	
DM_COM_Addin.cls	Private Sub DMWordAddinClose()
Word.DMClose	Dim rc As Boolean
	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	<pre>rc = oDMWordAddin.DMClose(ActiveDocument)</pre>
	lf (rc = False) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub
DM_COM_Addin.cls	Private Sub DMWordAddinCompare()
Word.DMCompare	Dim rc As Boolean
nononempore	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	<pre>rc = oDMWordAddin.dmCompare</pre>
	If (rc = True) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub
	End Sub

DM_COM_Addin.cls Word.DMCompareA ndMerge	Private Sub DMWordAddinCompareAndMerge() Dim rc As Boolean
nomerge	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	rc = oDMWordAddin.dmCompareAndMerge
	If (rc = True) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub
DM_COM_Addin.cls	Private Sub DMWordAddinExit()
Word.DMExit	Dim rc As Bool ean
	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	rc = oDMWordAddin.DMExit
	If (rc = True) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub

DM_COM_Addin.cls Word.DMFooter

DM_COM_Addin.cls Word.DMInsertFile

Pri vate	Sub DMWordAddinFooter()
Dim	rc As Bool ean
Dim	oDMWordAddin As DM_COM_Addin.clsWord
Set	oDMWordAddin = New DM_COM_Addin.clsWord
oDMV	VordAddin.DMFooter ActiveDocument
Set	oDMWordAddin = Nothing
End Sub	
Pri vate	Sub DMWordAddinInsertFile()
Dim	rc As Bool ean
DIM	oDMWordAddin As DM_COM_Addin.clsWord
Set	oDMWordAddin = New DM_COM_Addin.clsWord
rc =	= oDMWordAddin.dmlnsertFile
lf ((rc = True) Then
	Debug. Print "Success"
Else	0
	Debug.Print "Fail"
E	5
End	
Set	oDMWordAddin = Nothing

End Sub

DM_COM_Addin.cls Word.DMInsert	Private Sub DMWordAddinlnsertPicture() Dim rc As Boolean
Picture	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	rc = oDMWordAddin.dmlnsertPicture
	lf (rc = True) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub
DM_COM_Addin.cls	Private Sub DMWordAddinNativeSave()
Word.DMNative	Dim rc As Boolean
Save	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	rc = oDMWordAddin.DMNativeSave(ActiveDocument)
	lf (rc = True) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub

DM_COM_Addin.cls Word.DMNative	Private Sub DMWordAddinNativeSaveAS()
	Dim rc As Bool ean
SaveAs	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	<pre>rc = oDMWordAddin.DMNativeSaveAs(ActiveDocument)</pre>
	If (rc = True) Then
	Debug. Print "Success" El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub
DM COM Addin.cls	
DM COM Addin.cls	Private Sub DMWordAddinOpen()
DM_COM_Addin.cls Word.DMOpen	Private Sub DMWordAddinOpen() Dim rc As Boolean
	Dim rc As Boolean
	Dim rc As Boolean
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord rc = oDMWordAddin.DMOpen
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord rc = oDMWordAddin.DMOpen If (rc = True) Then
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord rc = oDMWordAddin.DMOpen If (rc = True) Then Debug.Print "Success"
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord rc = oDMWordAddin.DMOpen If (rc = True) Then Debug.Print "Success" Else
	Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord rc = oDMWordAddin.DMOpen If (rc = True) Then Debug.Print "Success" Else Debug.Print "Fail"
	<pre>Dim rc As Boolean Dim oDMWordAddin As DM_COM_Addin.clsWord Set oDMWordAddin = New DM_COM_Addin.clsWord rc = oDMWordAddin.DMOpen If (rc = True) Then Debug.Print "Success" Else Debug.Print "Fail" End If</pre>

DM_COM_Addin.cls Word.DMPassword ProtectDocument

DM_COM_Addin.cls Word.DMPrint

End Sub

DN COM Addia de	
DM_COM_Addin.cls Word.DMSave	Private Sub DMWordAddinSave()
MOIO'DWI29A6	Dim rc As Boolean
	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	<pre>rc = oDMWordAddin.DMSave(ActiveDocument)</pre>
	If (rc = True) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMWordAddin = Nothing
	End Sub
DM_COM_Addin.cls	Private Sub DMWordAddinSaveAS()
Word.DMSaveAS	Dim rc As Boolean
	Dim oDMWordAddin As DM_COM_Addin.clsWord
	Set oDMWordAddin = New DM_COM_Addin.clsWord
	rc = oDMWordAddin.DMSaveAS(ActiveDocument, "TEST", "TEST", 1)
	rc = oDMWordAddin.DMSaveAS(ActiveDocument, "TEST", "TEST", 1) If (rc = True) Then
	"TEST", 1)
	"TEST", 1) If (rc = True) Then
	"TEST", 1) If (rc = True) Then Debug.Print "Success"
	"TEST", 1) If (rc = True) Then Debug.Print "Success" Else
	"TEST", 1) If (rc = True) Then Debug.Print "Success" Else Debug.Print "Fail"
	"TEST", 1) If (rc = True) Then Debug.Print "Success" Else Debug.Print "Fail" End If

6

DM COM Add-in Programmer's Guide for Microsoft Excel

The DM COM Add-in exposes an automation interface with the following methods:

- DMClose
- DMCloseWorkspace
- DMExit
- DMFooter
- DMInsertPicture
- DMNativeSave
- DMNativeSaveAs
- DMOpen
- DMOpenVersion
- DMPwdProtDoc
- DMRestoreWorkspace
- DMSave
- DMSaveAS
- DMSaveWorkspace

DM_COM_ADDIN.clsExcel.DMClose

The **DMClose** function is used to release the DM reference to the opened document object.

Function DMClose(oDoc As Workbook) As Boolean

Parameters

oDoc

[in] Excel Workbook object.

Return Values

If the function succeeds, the return value will be VB_FALSE (0), and the document's status will be changed to Available. If the function fails, the return value will be VB_TRUE (-1).

Remarks

The **DMClose** function will not cause an application to close the actual file. **DMClose** is called by the Excel **WorkbookBeforeClose** event.

DM_COM_ADDIN.clsExcel.DMCloseWorkspace

The **DMCloseWorkspace** function is used to close all open workbooks that are part of the current workspace.

Function DMCloseWorkspace()

DM_COM_ADDIN.clsExcel.DMExit

The DMExit function is used to close all open workbooks and exit Excel.

Function DMExit() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1), and the document's status will be changed to Available. If the function fails, the return value will be VB_FALSE (0).

DM_COM_ADDIN.clsExcel.DMFooter

The **DMFooter** function is used to add a footer to the document that contains DM profile data.

Sub DMFooter(oDoc As Workbook)

Parameters

oDoc

[in] Excel Workbook object.

Remarks

This function inserts the Document Number, Document Name, Document Author, Document Type, Document Abstract, and Application ID.

DM_COM_ADDIN.clsExcel.DMInsertPicture

The **DMInsertPicture** function is used to insert a DM picture into the active workbook.

Function DMInsertPicture() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

You must select a valid picture type, such as .JPG.

DM_COM_ADDIN.clsExcel.DMNativeSave

The **DMNativeSave** function is used to save an existing DM or native document when using Passive Integration.

Function DMNativeSave(oDoc As Workbook) As Boolean

Parameters

ODoc

[in] Excel Workbook object.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

DM_COM_ADDIN.clsExcel.DMNativeSaveAs

The **DMNativeSaveAs** function is used to save as an existing DM or native document when using Passive Integration.

Function DMNativeSaveAs(oDoc As Workbook, [iDoClose As Integer]) As Boolean

Parameters

ODoc

[in] Excel Workbook object.

iDoClose

[in, optional] An integer, which specifies whether the document should be released from DM.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

If the document is a DM document, the document's status will be changed to Available.

DM_COM_ADDIN.clsExcel.DMOpen

The **DMOpen** function is used to open documents from DM.

Function DMOpen() As Boolean

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

DM_COM_ADDIN.clsExcel.DMOpenVersion

The **DMOpenVersion** function is used to open a specific version of a document.

Sub DMOpenVersion(iDocNumber As Long, sVersionID As String) $% \left({{{\rm{D}}_{{\rm{D}}}}_{{\rm{D}}}} \right)$

Parameters

iDocNumber

[in] The DM document number.

sVersionID

[in, optional] The DM Version ID.

DM_COM_ADDIN.clsExcel.DMPwdProtDoc

The DMPwdProtDoc function is used to display the Save With Options dialog box.

Sub DMPwdProtDoc()

DM_COM_ADDIN.clsExcel.DMRestoreWorkspace

The DMRestoreWorkspace function is used to open an Excel workspace.

Sub DMRestoreWorkspace()

Remarks

This function opens Excel workspaces that are saved to DM. (This should not be confused with DM Workspaces.)

DM_COM_ADDIN.clsExcel.DMSave

The **DMSave** function is used to save and upload the document to DM.

Function DMSave(oDoc As Workbook) As Boolean

Parameters

oDoc

[in] Excel Workbook object.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

If the document is not managed, Excel performs a native save operation.

DM_COM_ADDIN.clsExcel.DMSaveAS

The **DMSaveAS** function is used to create the Document Profile and obtain a return path to save the actual file.

Function DMSaveAS(oDoc As Workbook, [sProtectPass As String], [sWritePass As String], [ReadOnlyRec As Boolean], [CalledFromSave As Integer], [CalledFromNew As Integer]) As Boolean

Parameters

oDoc

[in] Excel Workbook object.

sProtectPass

[in,optional] Protection password to apply to workbook.

sWritePass

[in,optional] Write-Reserve password to apply to workbook.

ReadOnlyRec

[in,optional] Open workbook as read-only recommended.

CalledFromSave

[in,optional] Not used.

CalledFromNew

[in,optional] Not used.

Return Values

If the function succeeds, the return value will be VB_TRUE (-1). If the function fails, the return value will be VB_FALSE (0).

Remarks

The **DMSaveAS** function does not upload the file.

DM_COM_ADDIN.clsExcel.DMSaveWorkspace

The **DMSaveWorkspace** function is used to save an Excel DM workspace.

Sub DMSaveWorkSpace()

Remarks

This function saves Excel workspaces to DM. (This should not be confused with DM Workspaces.)

Sample Code

The sample code shown below was written in Microsoft Word VBA. To use these sample routines, you must include Office DM Integration Addin type library as a reference in your VB or VBA project.

DM_COM_ADDIN.	Private Sub DMExcel AddinClose()
clsExcel.DMClose	Dim rc As Boolean
	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	rc = oDMExcelAddin.DMClose(ActiveWorkbook)
	If (rc = False) Then
	Debug. Print "Success"
	El se
	Debug. Print "Fail"
	End If
	Set oDMexcel Addin = Nothing
	End Sub
DM_COM_ADDIN.	Private Sub DMExcelAddinCloseWorkspace()
clsExcel.DMClose	Dim rc As Boolean
Workspace	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	oDMExcel Addin. DMCI oseWorkSpace
	Set oDMExcel Addin = Nothing
	End Sub

DM_COM_ADDIN.cl sExcel.DMExit	Private Sub DMExcelAddinExit() Dim rc As Boolean Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	oDMExcel Addin. DMExit
	Set oDMExcelAddin = Nothing
	End Sub
DM_COM_ADDIN. clsExcel.DMFooter	Private Sub DMExcelAddinFooter() Dim rc As Boolean Dim oDMExcelAddin As DM_COM_Addin.clsExcel Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	oDMExcelAddin.DMFooter (ActiveWorkbook)
	Set oDMExcel Addin = Nothing
	End Sub
DM_COM_ADDIN. clsExcel.DMInsert Picture	Private Sub DMExcelAddinInsertPicture() Dim rc As Boolean Dim oDMExcelAddin As DM_COM_Addin.clsExcel Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	rc = oDMExcel Addin. DMI nsertPicture
	lf (rc = True) Then Debug.Print "Success" Else Debug.Print "Fail" End lf Set oDMExcelAddin = Nothing
	End Sub
DM_COM_ADDIN. clsExcel.DMNativeS ave	Private Sub DMExcelAddinNativeSave() Dim rc As Boolean Dim oDMExcelAddin As DM_COM_Addin.clsExcel Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	rc = oDMExcelAddin.DMNativeSave(ActiveWorkbook)

If (rc = True) Then Debug. Print "Success" El se Debug. Print "Fail" End If Set oDMExcel Addin = Nothing End Sub Private Sub DMExcel Addi nNati veSaveAs() Dim rc As Boolean Dim CloseDocinDM As Integer CloseDocinDM = 0Dim oDMExcelAddin As DM_COM_Addin.clsExcel Set oDMExcelAddin = New DM_COM_Addin.clsExcel rc = oDMExcel Addi n. DMNati veSaveAs(Acti veWorkbook, CloseDocinDM) If (rc = True) Then Debug. Print "Success" El se Debug. Print "Fail" End If Set oDMExcel Addin = Nothing

End Sub

DM_COM_ADDIN. clsExcel.DMNativeS aveAs

DM_COM_ADDIN. clsExcel.DMOpen	Private Sub DMExcelAddinOpen() Dim rc As Boolean
	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	rc = oDMExcel Addin.DMOpen
	lf (rc = True) Then Debug.Print "Success" Else Debug.Print "Fail"
	End If
	Set oDMExcel Addin = Nothing
	End Sub
DM_COM_ADDIN. clsExcel.DMOpen	Private Sub DMExcelAddinOpenVersion() Dim rc As Boolean
Version	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	oDMExcel Addin. DMOpenVersion 222, 3
	Set oDMExcel Addin = Nothing
	End Sub
DM_COM_ADDIN. clsExcel.DMPwd	Private Sub DMExcelAddinPwdProtDoc() Dim rc As Boolean
ProtDoc	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcel Addin = New DM_COM_Addin.clsExcel
	oDMExcel Addi n. DMPwdProtDoc
	Set oDMExcel Addin = Nothing
	End Sub
DM_COM_ADDIN. clsExcel.DM	Private Sub DMExcelAddinRestoreWorkspace() Dim rc As Boolean
RestoreWorkspace	Dim oDMExcelAddin As DM_COM_Addin.clsExcel

	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	oDMExcel Addin. DMRestoreWorkspace
	Set oDMExcel Addin = Nothing
	End Sub
DM_COM_ADDIN.	Private Sub DMExcel AddinSave()
clsExcel.DMSave	Dim rc As Boolean
	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	<pre>rc = oDMExcelAddin.DMSave(ActiveWorkbook)</pre>
	If (rc = True) Then
	Debug. Print "Success" El se
	Debug.Print "Fail"
	End If
	Set oDMExcelAddin = Nothing
	End Sub
DM_COM_ADDIN.	Private Sub DMExcel AddinSaveAS()
clsExcel.DMSaveAS	Dim rc As Boolean
	Dim oDMExcelAddin As DM_COM_Addin.clsExcel
	Set oDMExcelAddin = New DM_COM_Addin.clsExcel
	rc = oDMExcel Addin.DMSaveAS(ActiveWorkbook, "test", "test", True)
	If (rc = True) Then Debug.Print "Success"
	El se
	Debug.Print "Fail"
	End If
	Set oDMExcel Addin = Nothing
	End Sub

DM_COM_ADDIN. clsExcel.DMSave Workspace

Private Sub DMExcel Addi nSaveWorkspace() Dim rc As Boolean Dim oDMExcel Addin As DM_COM_Addin.clsExcel Set oDMExcel Addin = New DM_COM_Addin.clsExcel oDMExcel Addin.DMSaveWorkSpace Set oDMExcel Addin = Nothing

End Sub



Index

С

COM Automation for Microsoft Excel 27 COM Automation for Microsoft Word 5

D

DM Mail Merge 12 DM Simplified API Technical Specifications 49

F

Footers Microsoft Excel 34 Microsoft Word 16

I

Installing Excel COM Add-in 29 Word COM Add-in 7

Μ

Mail merge creating a data source in Word 15 Open a Native Data Source 15

Selecting a Saved Data Source 14 Word 12 Member Functions Excel DMClose 76 DMCloseWorkspace 76 DMExit 76 DMFooter 77 DMInsertPicture 77 DMNativeSave 77 DMNativeSaveAs 78 DMOpen 78 DMOpenVersion 79 DMPwdProtDoc 79 DMRestoreWorkspace 79 DMSave 79 DMSaveAS 80 DMSaveWorkspace 81

SAPI

DM.CloseDoc 51 DM.GetDocInfo 53 DM.IsDMDoc 53 DM.Library 54 DM.NewDoc 53 DM.SaveDoc 51 DM.SaveDocAs 52 Word DMClose 62 DMCompare 62 DMCompareAndMerge 62 DMExit 63 DMFooter 63 DMInsertFile 63 DMInsertPicture 64 DMNativeSave 64 DMNativeSaveAs 65 DMOpen 65 DMPasswordProtectDocument 65 DMPrint 66 DMSave 66 DMSaveAS 66 Microsoft applications Word mail merge 15 selecting a data source 14

Ν

New Functionality Microsoft Excel 28 Microsoft Word 6

0

open a native data source 15

S

Sample Code Excel 82 DMClose 82 DMCloseWorkspace 82 DMExit 83 DMFooter 83 DMInsertPicture 83 DMNativeSave 83 DMNativeSaveAs 84 DMOpen 85 DMOpenVersion 85 DMPwdProtDoc 85 DMRestoreWorkspace 85 DMSave 86 DMSaveAS 86 DMSaveWorkspace 87 SAPI DM.CloseDoc 55 DM.GetDocInfo 60 DM.IsDMDoc 59 DM.OpenDoc 50, 55 DM.SaveAs 58 DM.SaveDoc 57

Word DMClose 68 DMCompare 68 DMCompareAndMerge 69 DMExit 69 DMFooter 70 DMInsertFile 70 DMInsertPicture 71 DMNativeSave 71 DMNativeSaveAs 72 DMOpen 72 DMPasswordProtectDocument 73 DMPrint 73 DMSave 74 DMSaveAS 74 Supporting Software Microsoft Excel 28 Microsoft Word 6

W

Word mail merge 12